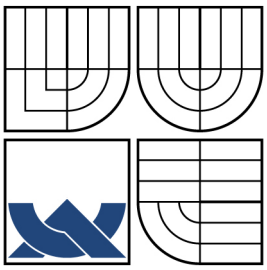
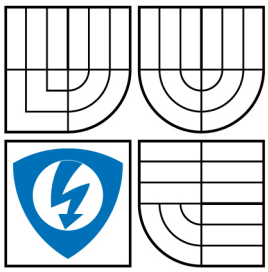


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘÍCÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

ANALÝZA SIGNÁLU
SIGNAL ANALYSIS

BAKALÁŘSKÁ PRÁCE
BACHELORS´S THESIS

AUTOR PRÁCE — PAVEL VÁVRA
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. PETR BENEŠ, Ph.D.
SUPERVISOR

BRNO 2008

ANOTACE

Práce se zabývá problematikou analýzy signálu, získaného měřením rovnoměrnosti chodu harmonických a planetových převodovek. V úvodní části jsou vysvětleny některé vlastnosti FFT a pojem kepstrum. Dále je uveden přehled nejčastěji měřených parametrů uvedených převodovek a potřebné vztahy, které vyjadřují souvislosti mezi jednotlivými mechanickými částmi převodovek. Další část se zabývá popisem SW modulu vytvořeného v prostředí LabView, který je výstupním produktem práce a provádí analýzu na změřených průbězích.

KLÍČOVÁ SLOVA

planetová převodovka, harmonická převodovka, rovnoměrnost chodu, mrtvý chod, analýza signálu

ANNOTATION

This thesis deals with problem of analysis signal acquiring measurement of uniformity motion harmonic drive and planetary gearboxes. In introduction part are explain some features of FFT and concept of cepstrum. Further is present overview of most often measured parameters of these gearboxes and needed formulas, which express connections among particular mechanical parts of gearboxes. The next part deals with description of SW program unit generated in LabView environment, which is output product of work and makes analysis on measured courses.

KEYWORDS

planetary gearbox, harmonic drive, uniformity motion, lost motion, signal analysis

Bibliografická citace

VÁVRA, P. *Analýza signálu.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 48 s. Vedoucí bakalářské práce doc. Ing. Petr Beneš, Ph.D.

P r o h l á š e n í

„Prohlašuji, že svoji bakalářskou práci na téma Analýza signálu jsem vypracoval samostatně, pod vedením vedoucího bakalářské práce, s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků, vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne : Podpis:

P o d ě k o v á n í

Děkuji tímto doc. Ing. Petru Benešovi, Ph.D. za cenné připomínky a rady při vypracování bakalářské práce.

V Brně dne : Podpis:

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

1. ÚVOD

Cílem práce bylo navrhnout SW modul v prostředí LabView pro analýzu harmonických složek v signálu, získaném měřením rovnoměrnosti chodu harmonických a planetových převodovek. Planetové a harmonické převodovky se stále více používají kvůli jejich větší přesnosti, menším výkonovým ztrátám a většímu převodovému poměru, oproti jednoduchým převodům. Proto je analýza jejich chování pro technickou praxi důležitá.

Analýzou průběhu vibrací nebo hluku měřeného na výstupní hřídeli planetové nebo harmonické převodovky můžeme usuzovat na vady jednotlivých zubů a na vady způsobené házením vstupního nebo výstupního hřídele, a nebo na stupeň opotřebení. Jelikož tato analýza nevyžaduje demontáž převodovky a nedochází při ní k žádnému stupni destrukce, je výhodná z několika hledisek. První je určitě časové hledisko. Taková analýza bude určitě rychlejší, než klasické analýzy vyžadující demontáž převodovky. Dalším hlediskem je jistě hledisko ekonomické, které je v dnešní době skoro nejdůležitější. Lze tento druh analýzy považovat za levnější, než klasické diagnostické metody. Odpadá totiž nutnost zaplatit pracovníka, který musí převodovku za účelem analýzy demontovat.

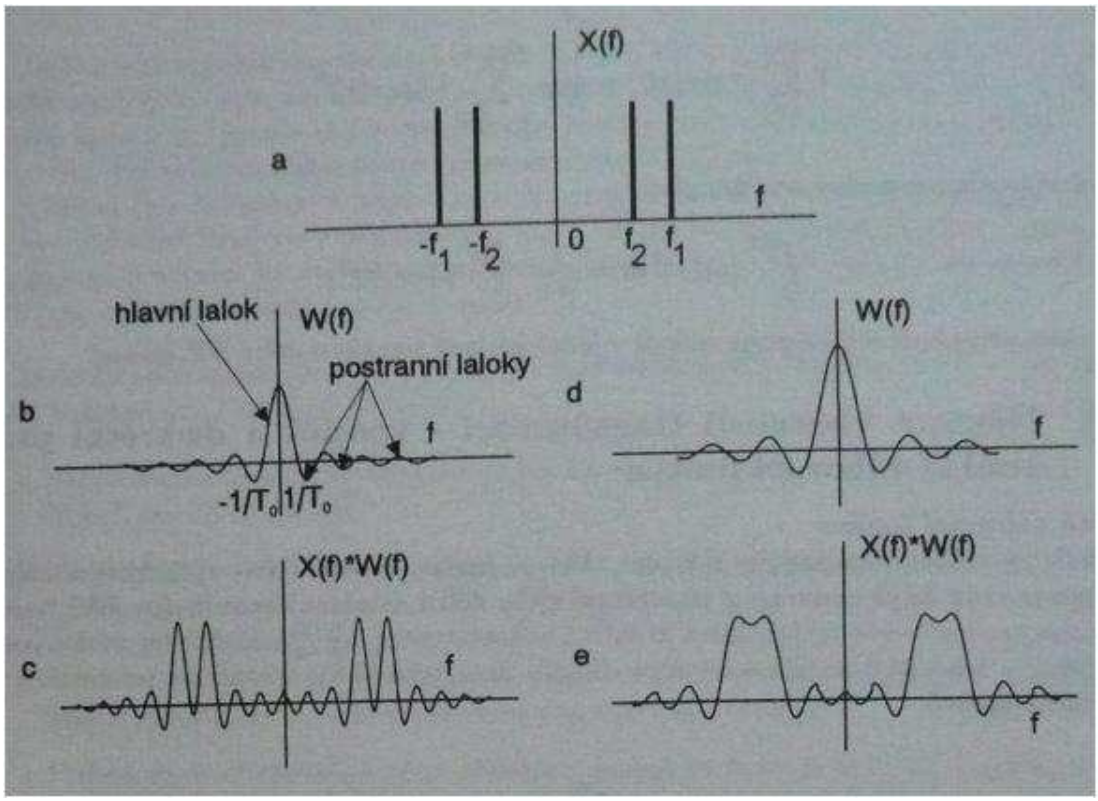
Programové prostředí LabView je pro analýzu signálů získaných měřením rovnoměrnosti chodu velice vhodné. Je to hlavně kvůli mnohým naprogramovaným virtuálním přístrojům a vestavěným funkcím. Tyto virtuální přístroje a funkce nám výrazně usnadňují vývoj takovýchto modulů, hlavně z pohledu času, věnovanému tvorbě modulu. LabView má také výborně zpracovanou nápovědu a systém příkladů k jednotlivým funkcím a virtuálním přístrojům. Dalším zdrojem informací při programování, který nelze opomenout, je internet. Na adrese http://www.ni.com/labview/ lze najít diskusní fóra, další užitečné příklady, virtuální přístroje menšího rozsahu a mnoho dalšího. Virtuální přístroje a příklady lze stáhnout, upravit a využít ve svých vlastních aplikacích.

2. SPEKTRÁLNÍ TRANSFORMACE

2.1 NĚKTERÉ VLASTNOSTI DISKRÉTNÍ FOURIEROVY TRANSFORMACE

2.1.1 Váhovací funkce

Pokud zaznamenáváme signál, který chceme analyzovat, pouze určitý omezený čas, je signál váhován obdélníkovým průběhem (časovým oknem), který tvar signálu nijak nemění. Tento obdélníkový průběh je funkce, která má funkční hodnotu 1 uvnitř intervalu $<-T_0/2;T_0/2>$, jinak je nulová[1]. V časové oblasti se vlastně jedná o vzájemné násobení sledovaného signálu a obdélníkového průběhu. Délka záznamu má zásadní vliv na výsledné frekvenční spektrum, jak ukazuje obrázek 2.1.



Obrázek 2.1 Vliv délky okna na tvar spektra

[1] Uhlíř, J., Sovka, P., Číslicové zpracování signálů, Vydavatelství ČVUT, Praha 1995, str.113-114

Na obrázku 2.1a je znázorněno spektrum signálu složeného ze dvou složek o frekvencích f_1 a f_2. Pokud tento signál budeme sledovat pouze omezený čas T_0, je výsledné spektrum dáno konvolucí (obraz násobení v časové oblasti je konvoluce ve frekvenční oblasti) spektra z obr. 2.1a a spektra obdélníkového průběhu, které je zobrazeno na obr. 2.1b. Z obr. 2.1b je vidět, že délka záznamu je nepřímo úměrná šířce hlavního laloku spektra obdélníkového signálu (tzn. čím kratší záznam, tím širší lalok). Výsledné spektrum je zobrazeno na obr. 2.1c. Je vidět, že lze ještě snadno rozlišit obě frekvenční složky.

Na obrázku 2.1d je zobrazeno frekvenční spektrum obdélníkového signálu o čtvrtinové délce trvání, než je znázorněno na obrázku 2.1b (tzn. signál byl zaznamenáván pouze čtvrtinu původní doby). Výsledné spektrum signálu s touto dobou záznamu je zobrazeno na obr. 2.1e. Zde je už vidět, že obě složky splynuly dohromady.

Lze učinit následující závěr: Vzdálenost čar ve spektru je:

$$\Delta f = f_1 - f_2 \qquad \textbf{Rovnice 2-1}$$

Je-li,

$$f_0 = \frac{1}{T_0} \qquad \textbf{Rovnice 2-2}$$

což je frekvenční rozlišení, menší než Δf, jsou obě čáry ve výsledném spektru rozlišitelné a nedochází ke ztrátě informace. Je-li nerovnost opačná, ke ztrátě informace dochází a nelze již přesně rozlišit obě frekvenční složky.

2.1.2 Spektrální rozptyl (spectral leakage)

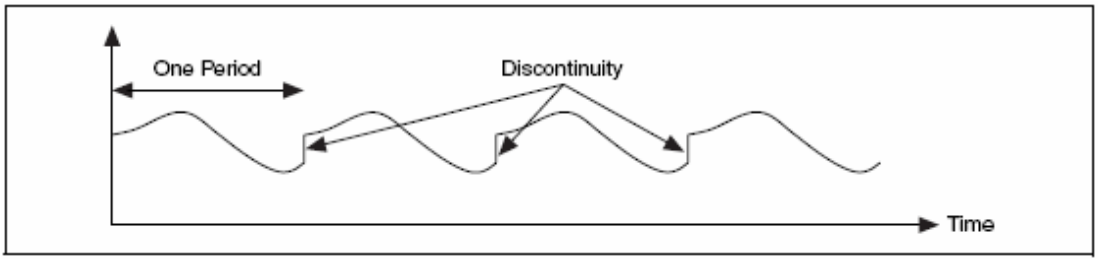
Spektrální rozptyl (spectral leakage) je další vlastnost diskrétní fourierovy transformace, která plyne z konečné doby záznamu signálu.

FFT algoritmus předpokládá, že se časový záznam sledovaného signálu přesně opakuje po celou dobu trvání signálu. Tzn. předpoklad, že časový záznam je celistvý násobek period původního signálu[2]. Pokud tomu tak není a délka záznamu odpovídá např. 0,85 násobku periody původního signálu, je FFT algoritmem

[2] National Instruments. Analysis Concepts. March 2004. p.5-1 - 5-5

vytvořen signál s nespojitostmi, které původní signál neobsahuje. Pro představu je uveden obrázek 2.2. Nespojitosti vytvořené uměle v signálu mají za následek výskyt frekvencí, které nejsou v původním signálu obsaženy a proto i výsledné spektrum tohoto signálu neodpovídá spektru původního měřeného signálu. Dochází ke spektrálnímu rozptylu.

Tomuto jevu se dá zabránit například synchronním vzorkováním s ohledem na měřený signál nebo nekonečnou dobou délky záznamu. Prvního se dá u některých



Obrázek 2.2 Průběh složený ze záznamu neobsahující celistvý násobek periody původního signálu

signálů snadno docílit - periodické signály. Avšak požadavek nekonečné délky záznamu je nerealizovatelný. Proto se někdy k omezení spektrálního rozptylu používá tzv. windowing. Princip spočívá v tom, že se časový záznam násobí časovými okny různých tvarů. Tyto tvary jsou vymyšleny tak, aby spektrální rozptyl co nejvíce potlačily.

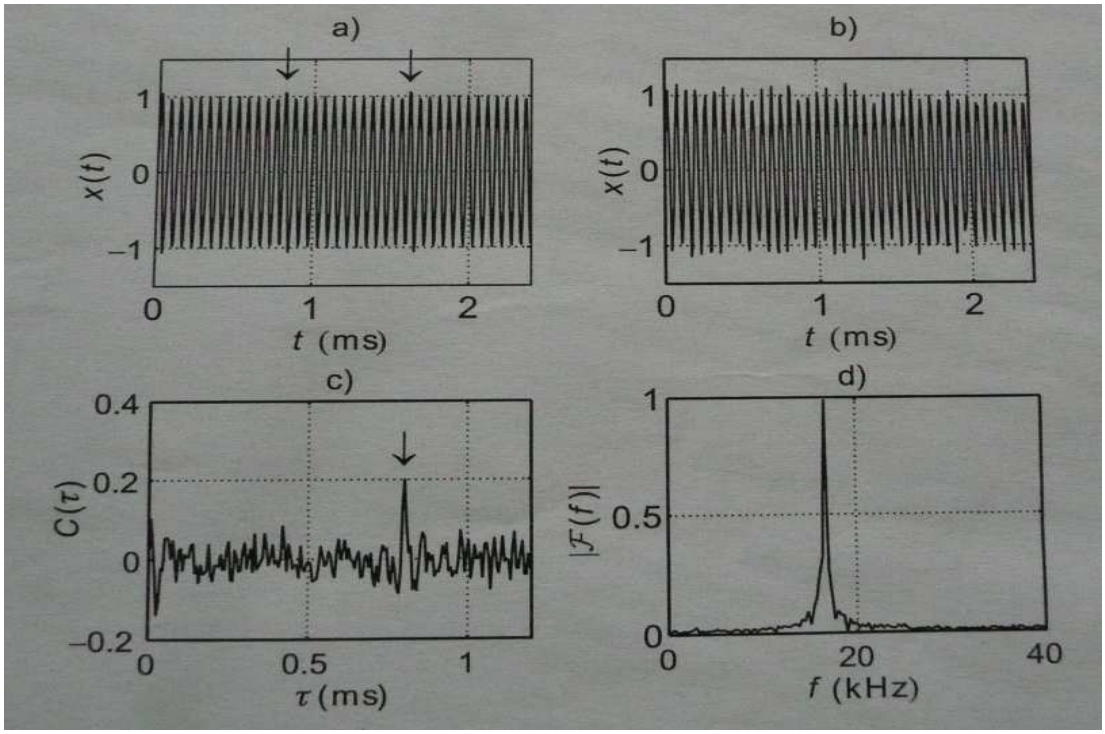
2.2 KEPSTRÁLNÍ ANALÝZA

Pojem kepstrum (vznik přesmyčkou prvních čtyř písmen ve slově spektrum) označuje metodu analýzy signálu, která aplikuje Fourierovu transformaci na logaritmické spektrum[3]. Svislá osa bývá většinou vyjádřena v dB a vodorovná osa, označovaná jako kvefrence (vznik opět přesmyčkou), má rozměr času. Tato transformace se vyznačuje vlastností citlivosti na ekvidistantně rozložené složky ve

[3] Kreidl, M., Šmíd, R., Technická diagnostika - senzory, metody, analýza signálu, BEN, 2006, ISBN 80-7300-158-6, str.56-59

spektru signálu, jako např. projev postranních pásem při amplitudové modulaci nebo skupin harmonických.

Pro představu použití kepstra je uveden obrázek 2.3. V části 2.3a je zobrazen harmonický signál o frekvenci 16,8 kHz, jehož amplituda je modulována obdélníkovým signálem s periodou 0,8ms. Na obr. 2.3b je stejný signál s přidaným normálně rozloženým bílým šumem. Zde už není projev modulace patrný (tento projev je na obr. 2.3a zvýrazněn šipkami). V kepstru, které je zobrazeno na obrázku 2.3c se modulační kmitočet projeví zřetelnou špičkou na kvefrenci odpovídající periodě modulace. Na obr. 2.3d je zobrazeno amplitudové spektrum zašuměného signálu, ve kterém nejsou postranní pásma amplitudové modulace zřetelná.



Obrázek 2.3 Kepstrální analýza

3. PARAMETRY A ANALÝZA SIGNÁLU PŘESNÝCH PŘEVODOVEK

3.1 PARAMETRY PŘESNÝCH PŘEVODOVEK

Mezi důležité měřené parametry přesných převodovek (tak bývají označovány planetové a harmonické převodovky) patří zejména rovnoměrnost chodu a mrtvý chod (angl. Lost motion) jak uvádí[4].

Rovnoměrnost chodu je parametr, který určuje odchylku skutečné pozice od ideální pozice výstupního hřídele při ideálním přenosu výkonu. V ideálním případě platí:

$$\Delta\varphi_2 = \frac{\Delta\varphi_1}{i_{id}} \qquad \textbf{Rovnice 3-1}$$

kde $\Delta\varphi_2$ resp. $\Delta\varphi_1$ - změna úhlu výstupního(vstupního) hřídele [rad]

i_{id} - převodový poměr dané převodovky daný počtem zubů

V důsledku nepřesností při výrobě a opotřebením lze při dostatečně dlouhém měření polohy výstupního a vstupního hřídele vynést závislost:

$$v(\varphi_2) = \varphi_2 - \frac{\varphi_1}{i_{id}} \qquad \textbf{Rovnice 3-2}$$

Tímto získáme úhlovou resp. časovou závislost převodového poměru. Pokud na takovýto průběh aplikujeme FFT, můžeme identifikovat složky spektra, které přísluší jednotlivým ozubeným kolům nebo hřídelím. Rovnoměrnost chodu se měří při malých otáčkách, kvůli potlačení dynamických sil působících na převodovku. Při měření při nízkých otáčkách se dá měření považovat za statické. Rovnoměrnost chodu se měří v obou směrech otáčení výstupního hřídele (napravo a nalevo), protože se průběhy liší. Je to způsobeno nepřímým tvarem ozubení.

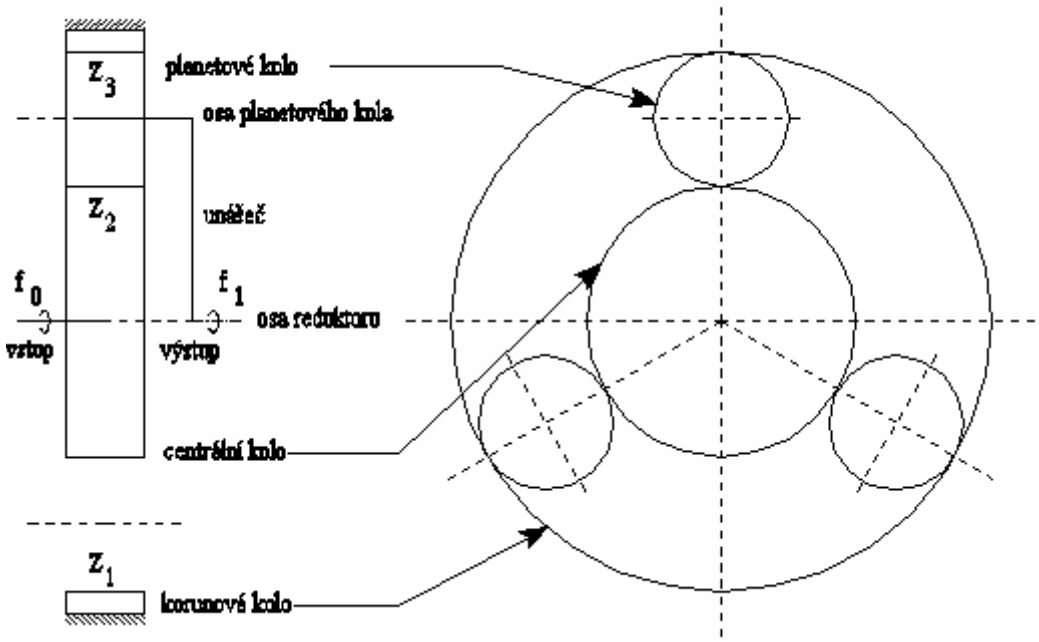
---

[4] Malec, Z., Beneš, P., Klusáček, S. Vývoj metod pro měření parametrů přesných převodovek : závěrečná zpráva projektu GA ČR 102/01/ 1044. Brno: 10.1. 2004

Druhým parametrem je mrtvý chod. Tento jev se dá vysvětlit jako klasická nelinearita vůle v převodech. Zuby na sobě nesedí vždy úplně natěsno a proto při reverzaci (změně směru chodu) dochází k pohybu vstupního hřídele, který nemá na výstupní hřídel, co se týče přenášeného výkonu žádný vliv. Pokud je měření rovnoměrnosti chodu vlevo a vpravo prováděno přesně ve stejných měřících bodech po obvodu hřídele, dá se průběh mrtvého chodu spočítat prostým odečtením těchto dvou průběhů.

3.2 ANALÝZA SIGNÁLU Z PLANETOVÉ PŘEVODOVKY

Rozbor spektra u planetových převodovek je oproti jednoduchým soukolím složitější. Planetové kolo je totiž v záběru nejen s centrálním a korunovým kolem, ale jeho osa se otáčí společně s unášečem. Proto je spektrum daleko bohatší.



Obrázek 3.1 Kinematické schéma planetového převodu z knihy[5]

Kinematické schéma jednoho stupně planetové převodovky je na obrázku 3.1. V tomto schématu je korunové kolo pevně uchyceno k převodové skříni. Počet zubů

[5] Tůma, J. Zpracování signálů získaných z mechanických systémů užitím FFT „1.vydání“, Sdělovací technika, Praha 1997, ISBN 80-901936-1-7, str.142

z_3 značí počet zubů planetového kola, z_2 počet zubů centrálního kola a z_1 počet zubů korunového kola. Počet planetových kol je označen n. Frekvence otáčení vstupního hřídele je f_0 a frekvence otáčení výstupního hřídele je f_1. K analýze frekvenčního spektra potřebujeme převodový poměr. Ten je roven následujícímu vztahu:

$$\frac{f_1}{f_0} = \frac{z_2}{z_1 + z_2}$$ **Rovnice 3-3**

Tabulka 3-1 Vzájemný přepočet frekvencí periodicit u planetové převodovky

f_0	f_0	$\frac{z_1+z_2}{z_2} f_1$	$\frac{1}{n}\frac{z_1+z_2}{z_2} f_2$	$\frac{1}{n}\frac{z_1+z_2}{z_1} f_3$	$\frac{z_3}{z_1}\frac{z_1+z_2}{z_2} f_4$
f_1	$\frac{z_2}{z_1+z_2} f_0$	f_1	$\frac{1}{n} f_2$	$\frac{1}{n}\frac{z_2}{z_1} f_3$	$\frac{z_3}{z_1} f_4$
f_2	$n\frac{z_2}{z_1+z_2} f_0$	nf_1	f_2	$\frac{z_2}{z_1} f_3$	$n\frac{z_3}{z_1} f_4$
f_3	$n\frac{z_1}{z_1+z_2} f_0$	$n\frac{z_1}{z_2} f_1$	$\frac{z_1}{z_2} f_2$	f_3	$n\frac{z_3}{z_2} f_4$
f_4	$\frac{z_1}{z_3}\frac{z_2}{z_1+z_2} f_0$	$\frac{z_1}{z_2} f_1$	$\frac{1}{n}\frac{z_1}{z_3} f_2$	$\frac{1}{n}\frac{z_2}{z_3} f_3$	f_4
f_5	$\frac{z_1 z_2}{z_1+z_2} f_0$	$z_1 f_1$	$\frac{z_1}{n} f_2$	$\frac{z_2}{n} f_3$	$z_3 f_4$
f_6	$n\frac{z_1 z_2}{z_1+z_2} f_0$	$nz_1 f_1$	$z_1 f_2$	$z_2 f_3$	$nz_3 f_4$
f_7	$n\left[\frac{z_1}{n}\right]\frac{z_2}{z_1+z_2} f_0$	$n\left[\frac{z_1}{n}\right] f_1$	$\left[\frac{z_1}{n}\right] f_2$	$\left[\frac{z_1}{n}\right]\frac{z_2}{z_1} f_3$	$n\left[\frac{z_1}{n}\right]\frac{z_2}{z_1} f_4$

[…] symbol matematické operace “celá část čísla v závorce”

Frekvence lokálních poruch u planetové převodovky jsou následující:

- f_0 - otáčení centrálního kola, tj.vstupního hřídele
- f_1 - otáčení unášeče, tj.výstupního hřídele
- f_2 - dotyk zvoleného zubu korunového kola s planetovými koly
- f_3 - dotyk zvoleného zubu centrálního kola s planetovými koly

- f_4 - dotyk zvoleného zubu planetového kola buď s korunovým kolem nebo s centrálním kolem
- f_5 - frekvence dotyku zubů korunového kola se zuby jednoho planetového kola
- f_6 - frekvence dotyku zubů korunového kola současně se všemi planetovými koly
- f_7 - frekvence dotyku zubů korunového kola postupně se všemi planetovými koly

Prvních pět frekvencí je vhodných k hodnocení časového záznamu hluku nebo vibrací, které jsou vybuzeny záběrem ozubených kol. Další tři frekvence, které jsou daleko vyšší souvisí se záběrovým cyklem jednotlivých zubů všech ozubených kol.

Přehled vzorců pro výpočet všech uvedených frekvencí f_0 až f_7 pomocí prvních pěti základních frekvencí f_0 až f_4 je uveden v tabulce 3-1.

3.3 ANALÝZA SIGNÁLU Z HARMONICKÉ PŘEVODOVKY

Harmonické převodovky mají oproti planetovým převodovkách menší rozměry a vyšší účinnost. Jsou také schopny dosahovat velkého rozsahu převodových poměrů při stejném rozměru. Kinematické schéma harmonického převodu je na obrázku 3.2. Obrázek jsem stáhl z internetu[6] a popsal.

Pokud je převodovka uspořádána tak, že plní funkci reduktoru otáček (tzn. vstupní hřídel je připojena přes eliptický generátor vln, výstupní hřídel je připojena na pružné ozubené kolo a tuhé ozubené kolo je zafixováno a nepohybuje se) je převodový poměr roven následujícímu vztahu[7]:

$$i = \frac{z_{pruzne}}{z_{pruzne} - z_{tuhe}} \qquad \textbf{Rovnice 3-4}$$

kde i - převodový poměr

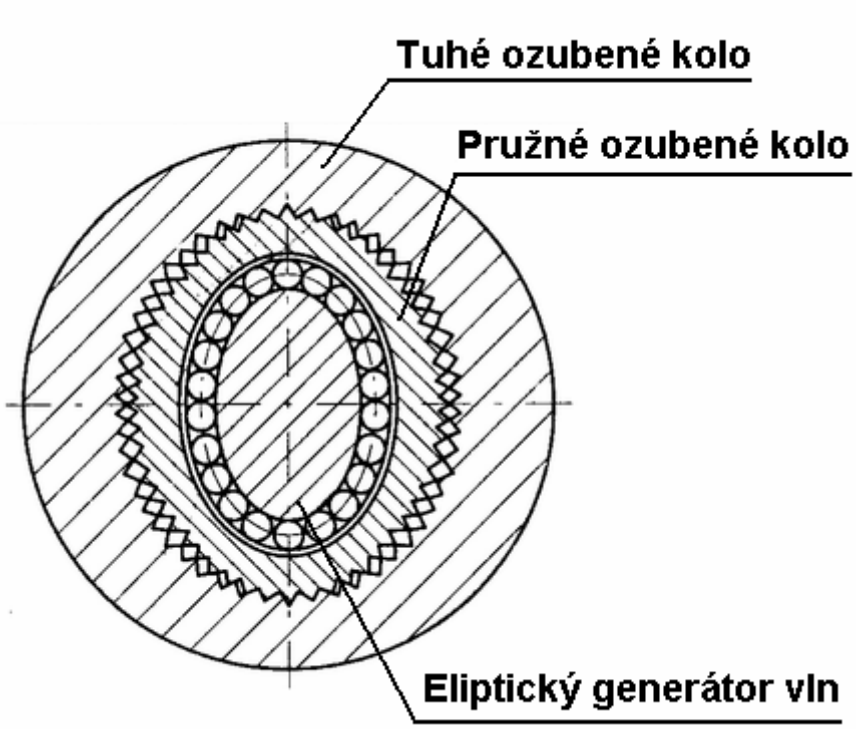
z_{pruzne} - počet zubů pružného ozubeného kola

z_{tuhe} - počet zubů tuhého ozubeného kola

Z tohoto převodového poměru by se dali odvodit podobné periodicity jako u planetové převodovky uvedené v tabulce 3-1.

[6] http://www.pavouk.net/users/ovitek/a00108f00001-www.volny.cz-ovitek-harmprev.gif

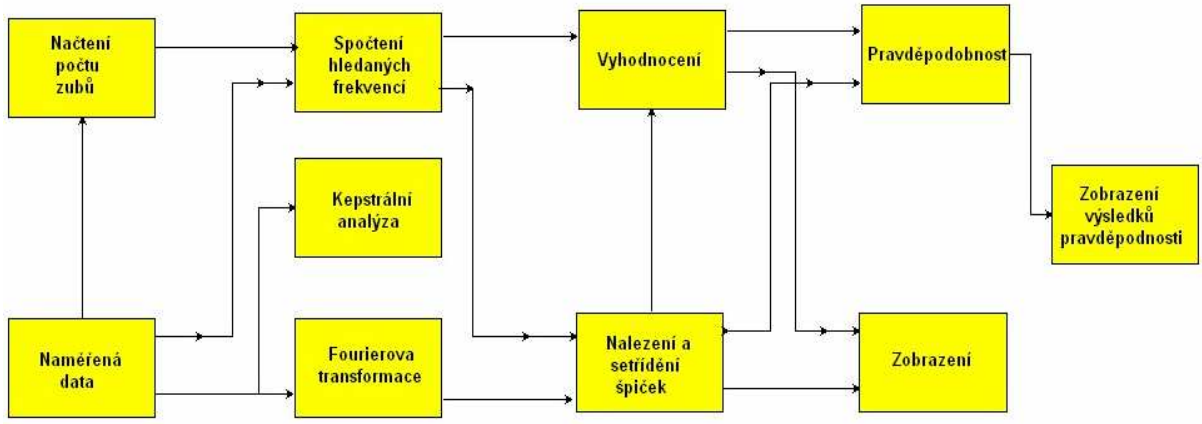
[7] http://www.harmonicdrive.net/reference/operatingprinciples/



Obrázek 3.2 Kinematické schéma harmonického převodu

4. ŘEŠENÍ SOFTWAROVÉHO MODULU

Softwarový modul pro analýzu průběhů z planetových převodovek je vytvořen v prostředí LabView 7.1. Modul pro analýzu z harmonických převodovek by byl téměř totožný. Musel by se změnit pouze blok **Spočítání hledaných frekvencí**. Pokud by však počty zubů převodovky byli v databázi uspořádány jinak než u planetových převodovek, změnu by vyžadoval i blok **Načtení počtu zubů**. Ostatní bloky by zůstaly stejné. Zjednodušené blokové schéma modulu je uvedeno na obrázku 4.1.



Obrázek 4.1 Blokové schéma softwarového modulu

Nyní budou jednotlivé bloky popsány.

4.1 NAMĚŘENÁ DATA

- vstupy
  a) file path (dialog if empty) - cesta k souboru s naměřenými daty
- výstupy
  a) new file path (Not A Path if cancelled) - absolutní cesta k souboru s naměřenými daty, která se budou analyzovat
  b) array of revolution - jednorozměrné pole s hodnotami zlomků otáček, ve kterých byla hodnota rovnoměrnosti chodu a mrtvého chodu snímána
  c) uniformity left - time course - průběh rovnoměrnosti chodu vlevo

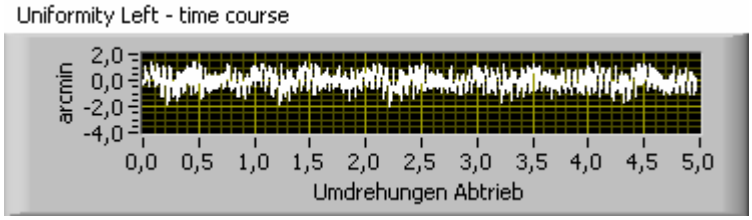
d) uniformity right - time course - průběh rovnoměrnosti chodu vpravo

e) lost motion - time course - průběh mrtvého chodu

Tento blok jsem neprogramoval, pouze jsem ho převzal od docenta Beneše a upravil. Úprava spočívala ve vymazání částí tohoto bloku, které nejsou pro moji aplikaci potřeba.

Vstupem do tohoto bloku je cesta k souboru s naměřenými daty, která budou analyzována. Jelikož je pole pro vpisování cesty na čelním panelu aplikace skryto (položka Hide Control v menu příslušného ovladače (control)), zobrazí LabView po spuštění aplikace souborový dialog box. To zajistí virtuální přístroj **Read Waveform from File.vi.** V tomto dialog boxu je stromová struktura všech dostupných disků, proto je snadné, dostat se k požadovanému souboru. Poté je potřeba z načteného souboru separovat potřebná data, abychom s nimi mohli dále pracovat.

Nejprve se ve smyčce **For Loop** uloží naměřené hodnoty jednotlivých průběhů pomocí funkce **Get Waveform Components** do dvourozměrného pole a poté se ve funkci **Index Array** separují do jednorozměrných polí. Tato jednorozměrná pole jsou spolu s cestou k souboru, ze kterého se právě analyzují data (cesta je použita jako vstup do bloku **Načtení počtu zubů**), výstupem z tohoto bloku. Získané průběhy rovnoměrnosti chodu a mrtvého chodu, jsou také zobrazeny na čelním panelu aplikace, jak ukazuje obrázek 4.2.



Obrázek 4.2 Průběh rovnoměrnosti chodu vlevo

4.2 FOURIEROVA TRANSFORMACE

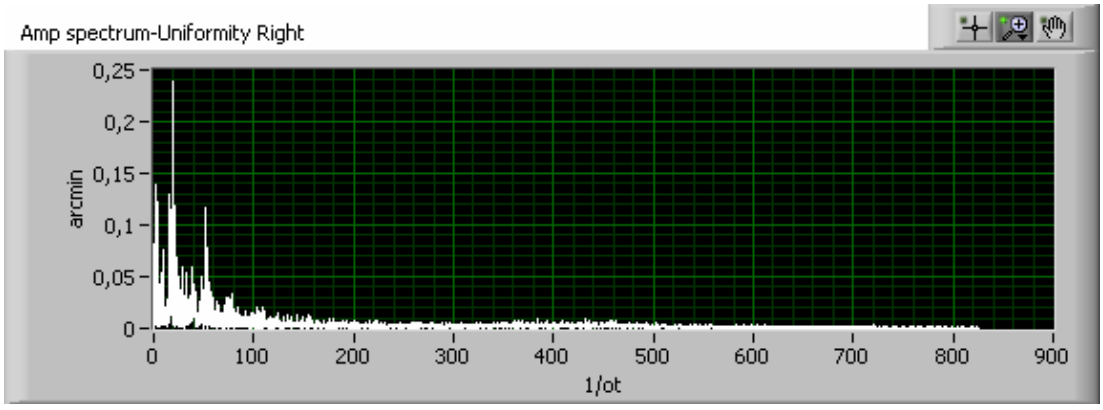
- použité vstupy

a) Signal (V) - průběh, který chceme převést do frekvenční oblasti

b) dt - vzorkovací perioda

- použité výstupy
    a) Amp Spectrum Mag (Vrms) - amplitudové spektrum
    b) Amp Spectrum Phase (radians) - fázové spektrum
    c) df - frekvenční rozlišení

Tento blok jsem programovat nemusel, použil jsem vestavěný virtuální přístroj **Amplitude and Phase Spectrum.vi**. Vstupem je signál, který chceme analyzovat (v našem případě průběhy získané ze souboru) a vstup dt**,** což je vzorkovací perioda. Vzorkovací periodu jsem získal odečtením prvního a nultého prvku jednorozměrného pole zlomků otáček, ve kterých byla měřena rovnoměrnost chodu nebo mrtvý chod. Toto pole je analogií pole s časovou posloupností. Jako výstup tento blok vrací posloupnost amplitud a fázi ve frekvenční oblasti a rozlišení frekvenčního spektra df. Výstupní posloupnost ve frekvenční oblasti má oproti vstupní posloupnosti poloviční velikost. Je to z toho důvodu, že se předpokládá vzorkovací frekvence alespoň dvakrát větší, než je největší frekvenční složka vzorkovaného signálu. V našem případě má vstupní posloupnost 8192 vzorků a výstupní posloupnost 4096 vzorků. Ukázka spočítaného amplitudového frekvenčního spektra je na obrázku 4.3.



Obrázek 4.3 Amplitudové frekvenční spektrum

4.3 KEPSTRÁLNÍ ANALÝZA

- použité vstupy
    a) Xt- průběh, na kterém chceme provést kepstrální analýzu
    b) sampling rate - vzorkovací frekvence

- použité výstupy

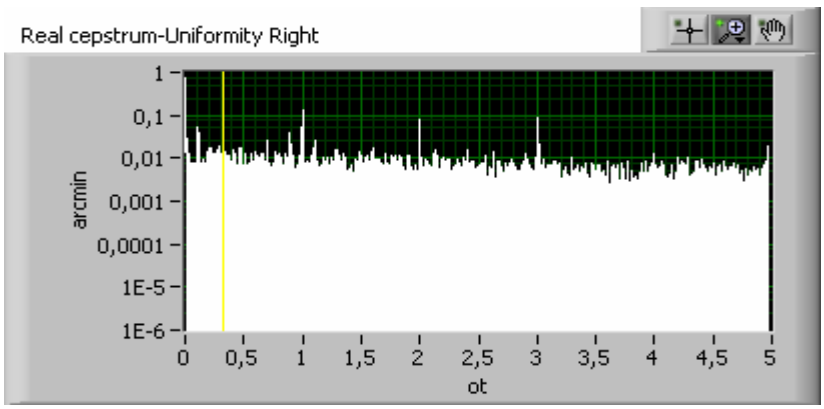
a) cepstrum - jednostranné reálné kepstrum

Kepstrální analýza je prováděna na průbězích rovnoměrnosti chodu a mrtvého chodu, za účelem nalezení periodicit v těchto průbězích. Byla snaha z této analýzy určovat pravděpodobnost příslušnosti složek spektra jednotlivým mechanickým zdrojům. Nakonec tato metoda nevedla k předpokládaným výsledkům.

Kepstrální analýzu provádí virtuální přístroj **TSA Real Cepstrum.vi**, který je součástí balíku Advanced Signal Processing Toolkit. Patří do skupiny Time Series Analysis VIs a podskupiny Correlation and Spectral Analysis VIs. Na vstup *Xt* tohoto virtuálního přístroje musím přivést průběh, na kterém chci provést kepstrální analýzu. Pokud je tento průběh ve formě jednorozměrného pole, musím specifikovat další vstup *sampling rate*, což je vzorkovací frekvence. Vzorkovací frekvence se spočítá podle vzorce:

$$f_s = \frac{1}{dt} \qquad \textbf{Rovnice 4-1}$$

kde dt je vzorkovací perioda. Proto musím nejprve vypočítat vzorkovací periodu, a poté zjistit její převrácenou hodnotu. Vzorkovací periodu získám stejným způsobem jako v předcházející kapitole odečtením prvního a nultého prvku jednorozměrného pole zlomků otáček, ve kterých byla měřena rovnoměrnost chodu nebo mrtvý chod. Výstupem je pak jednostranné reálné kepstrum, které je zobrazeno na čelním panelu, jak ukazuje obrázek 4.4. Vodorovná osa má rozměr otáček (stejně jako analyzovaný průběh) a svislá osa je vyjádřena v dB (viz kapitola 2.2).



Obrázek 4.4 Jednostranné reálné kepstrum

4.4 NAČTENÍ POČTU ZUBŮ

- vstupy
    a) path of .udl file - cesta k .udl souboru, který obsahuje informace o spojení s databází, ke které se chceme připojit
    b) path of file of measuring data - cesta k souboru s průběhy, které se právě analyzují
- výstupy
    a) array of teeth - jednorozměrné pole s počty zubů z databáze

Pro analýzu změřených průběhů potřebujeme znát počet zubů jednotlivých ozubených kol v převodovce. Počty zubů jsou uloženy v databázi, ve které by bylo ruční vyhledávání velmi obtížné a nepřehledné, a značně by analýzu zpomalovalo. Proto je modul vybaven tímto blokem, který otevře databázi, načte z ní počet zubů a vrátí je ve formě jednorozměrného pole. Blok je vytvořen za pomocí virtuálních přístrojů z Database Connectivity Toolset.

Nejprve musíme vytvořit spojení mezi naší aplikací vytvořenou v LabView a databází, ve které jsou uloženy počty zubů, jak uvádí[8] (tento dílčí úkon zpravidla bývá největším kamenem úrazu při práci s databázemi, o čemž jsem se přesvědčil i já na vlastní kůži). Pro tento účel musíme vytvořit informace o spojení s databází (její jméno, cesta k ní, ovladač, který zabezpečuje komunikaci mezi aplikací a databází). Můžeme tak udělat pomocí .dsn (data source name) nebo .udl (universal data link) souboru. Oba dva způsoby jsou dost podobné. Já jsem si vybral způsob s .udl souborem, protože lze tento soubor vytvořit přímo v LabView. Nyní uvádím krok za krokem postup, vytvoření .udl souboru.

V prostředí LabView v horní liště klepneme na **Tools»Create Data Link** a otevře se nám dialog box s názvem Vlastnosti Data Link. Nejprve musíme vybrat Zprostředkovatele OLE DB mezi aplikací a databází. Je na výběr z mnoha možností, záleží jakou máme databázi. My máme databázi v aplikaci MS Access, proto vybereme **Microsoft Jet 4.0 OLE DB Provider** a klepneme na tlačítko **Další**. Tímto úkonem se přejde na následující záložku Připojení. Zde vybereme databázi, ke které

[8] National Instruments. Database Connectivity Toolset User Manual. May 2001. p.3-1

se chceme připojit. V případě nutnosti vyplníme i další informace (uživatelské jméno a heslo). Poté, co vybereme databázi, můžeme zkusit otestovat připojení pomocí tlačítka **Testovat připojení**. Pokud se připojení zdařilo, můžeme postupovat dál. Zmáčkneme tlačítko **OK.** Nyní už musíme jen zadat název a umístění .udl souboru, na který budeme později odkazovat.

Jeden ze vstupů do bloku, jak je uvedeno na začátku této podkapitoly, je path of .udl file (cesta k .udl souboru). Není nastavena konstantní cesta k tomuto souboru. Počítal jsem s tím, že pokud si tento program někdo nahraje, bude vždy cesta k tomuto souboru trochu jiná a bude jiné i umístění databáze, ze které se zuby načítají. Proto se před prvním spuštěním aplikace nejprve vytvoří .udl soubor (podle návodu z předchozího odstavce), který musí splňovat dvě podmínky, aby program fungoval. Musí se jmenovat databaze.udl a musí být uložen ve stejném adresáři, jako je umístěn hlavní virtuální přístroj aplikace **Vavra_read_lvw.vi**. Tyto podmínky jsou odvozeny podle konstrukce cesty, která je přivedena právě na vstup path of .udl file.

Funkce **Propery Node** hlavního virtuálního přístroje **Vavra_read_lvw.vi** vrací aktuální cestu k tomuto VI (je použita jako vstup *string* do následující funkce **Search and Replace String** - dále **SaRS**) a název VI (je použit jako vstup *search string* do **SaRS**). Jako třetí vstup *replace string* do **SaRS** je použit prázdný řetězec, proto blok **SaRS** (který najde v řetězci na vstupu *string* řetězec, který je připojen na vstup *search string* a nahradí jej řetězcem, který je připojen na *replace string*) vrací cestu k adresáři, ve kterém je **Vavra_read_lvw.vi** umístěn. S touto cestou se ve funkci **Build Path** sloučí řetězec “databaze.udl“. Tím pádem **Build Path** vrací cestu k .udl souboru, která je použita jako vstup path of .udl file, pokud jsou splněny obě podmínky kladené na .udl soubor uvedené výše. Potom by spojení s databází mělo být bezproblémové.

Druhým vstupem do právě popisovaného bloku, je path of file of measuring data. Do tohoto vstupu je připojen výstup new file path (Not A Path if cancelled) z bloku **Naměřená data,** jak je o tom zmiňováno v kapitole 4.1. Tato cesta nám slouží ke zjištění, z jakého typu převodovky pocházejí naměřená data, která právě analyzujeme. Názvy souborů s naměřenými daty mají následující formát:

xxxx_yyyy.lvw, a nebo xxxx_zzz.lvw (xxxx a yyyy jsou čtyřmístná čísla, zzz je trojmístné číslo). Právě pomocí čtyřmístného čísla xxxx se v databázi dostaneme k převodovce, ze které data pocházejí. Proto musíme nejprve toto číslo z cesty separovat, abychom ho mohli využít pro následné hledání v databázi. Provedeme to pomocí funkcí z knihovny Array Functions.

Nejprve musíme cestu k souboru s naměřenými daty převést na řetězec (string), abychom mohli funkce z dané knihovny využít. Použijeme k tomu funkci **Path to String**. Dále využijeme toho, že ať už je soubor uložen kdekoliv tzn. jakkoliv složitá adresářová struktura, jsou jednotlivé adresáře odděleny v cestě zpětným lomítkem. Poslední zpětné lomítko v cestě bude vždy před samotným názvem souboru. Proto pomocí funkce **Search and Replace String** (dále **SaRS**) nalezneme umístění posledního zpětného lomítka v cestě (řetězci). Abychom toho dosáhli, musíme na vstupy funkce **SaRS** přivést následující: *string* - cesta k souboru s naměřenými daty převedená na řetězec; *search* string - zpětné lomítko \; *replace* string - zpětné lomítko \; replace *all?* - true. Vstupy *search a replace string* jsou stejné z důvodu toho, že nechceme řetězec měnit. Chceme pouze najít výskyt posledního zpětného lomítka v cestě. Pokud budou takto připojeny vstupy na výstupu *result string*, z funkce **SaRS** bude stejný řetězec jako na vstupu *string* a na výstupu *offset past replacement* bude index výskytu posledního zpětného lomítka. Dále použijeme funkci **Search/Split String** (dále **SSS**). Na vstup *string* přivedeme nezměněnou cestu, na vstup *offset* přivedeme index posledního výskytu zpětného lomítka v cestě. Potom nám funkce **SSS** rozdělí cestu na cestu k adresáři, kde je soubor uložen (výstup *substring before match* - tu nebudeme potřebovat) a na řetězec samotného názvu souboru (výstup *match + rest of string*) ve formátu xxxx_yyyy.lvw nebo xxxx_zzz.lvw. V názvu souboru je čtyřmístné číslo xxxx, které nás zajímá, odděleno podtržítkem od zbytku názvu. Toho využijeme a opět použijeme funkci **SSS**. Na vstup *string* přivedeme název souboru, na vstup *search string/char* přivedeme znak podtržítka. Potom budeme mít na výstupu *substring before match* kýžené čtyřmístné číslo, se kterým můžeme dále pracovat.

Nyní už máme všechny podklady k tomu, abychom získali počty zubů z databáze. Nejprve musíme databázi otevřít. To provedeme pomocí **DB Tools Open**

Connection.vi. Na vstup *connection information* připojíme vstup path of .udl file (cesta k .udl souboru). Jiný vstup u tohoto virtuálního přístroje není využit. Jako výstup dostaneme *connection reference*, který používáme jako vstup do dalších VI.

K počtům zubů se musíme propracovat přes tři tabulky v databázi. Pomocí čtyřmístného čísla, které jsme získali z cesty k aktuálně analyzovanému souboru a které je umístěno v tabulce s názvem TbdMeasResult, ve sloupci ID_MeasResult, získáme první číslo, které je umístěno ve stejné tabulce, ve stejném řádku jako získané číslo a ve sloupci ID_Order. Pomocí prvního čísla, které je dále v tabulce TbdOrders a ve sloupci ID_Order, získáme druhé číslo, které je opět ve stejné tabulce, řádku a ve sloupci ID_TypeGearBox. Nakonec pomocí druhého čísla umístěného v tabulce TbdGearBox a ve sloupci ID_TypeGearBox, získáme počty zubů jednotlivých ozubených kol. Tyto počty jsou ve stejném řádku jako druhé číslo a ve sloupcích GB_Z11, GB_Z12, GB_Z13, GB_Z21, GB_Z22, GB_Z23, GB_Z31, GB_Z32 a GB_Z33. Pokud některý ze sloupců GB_Zxy obsahuje 1, znamená to, že převodovka x-tý stupeň neobsahuje, s čímž se počítá v následném bloku **Spočítání hledaných frekvencí**.

Popsaná vyhledávací procedura se provede ve třech krocích, které jsou principiálně naprosto stejné. Je použit virtuální přístroj **DB Tools Select Data.vi** a z něho následující vstupy: vstup *connection reference* - na tento vstup je připojen výstup *connection reference* z **DB Tools Open Connection.vi**; vstup *table* - název tabulky v databázi, ve které budeme data hledat; vstup *columns* - název sloupců, ve kterých budeme data hledat a poslední vstup *optional clause* - na tento vstup můžeme přivést SQL příkaz, kterým filtrujeme výběr položek z daného sloupce. Proto vždy v každém ze tří kroků přivedeme název tabulky a sloupců, ze kterých chceme získat data. Na vstup *optional clause* přivedeme příkaz v následujícím tvaru: *where název_sloupce = získané_číslo*, kde: *název_sloupce* - je název sloupce, kde je umístěno předchozí získané číslo; *získané číslo* - je předchozí získané číslo. Toto nám zajistí výběr dat pouze ze stejného řádku, kde je umístěno v tabulce předchozí získané číslo (ať už čtyřmístné číslo získané z cesty a nebo číslo získané z předchozí tabulky). SQL příkaz je vždy sestaven pomocí funkce **Concatenate Strings**. Tato funkce slučuje libovolný počet řetězců v jeden řetězec. V našem případě slučuje dva

řetězce. Jako první je připojen řetězec ve tvaru *where název_sloupce* = (např. *where ID_MeasResult* =). Jako druhý je připojen řetězec obsahující číslo, které bylo získáno z předchozí tabulky, a nebo již několikrát zmiňované čtyřmístné číslo, získané z cesty k aktuálnímu analyzovanému souboru. Virtuální přístroj **DB Tools Select Data.vi** vrací na výstupu *connection reference out* odkaz na databázi, ve které vyhledával (použije se pro další vyhledávání nebo pro uzavření databáze) a na výstupu *data* data, která vyhledal v dané tabulce. Tato data už jsou přímo nalezené počty zubů nebo čísla, která se použijí k dalšímu vyhledávání v tabulce. Jsou však v datovém typu "database variants" , vždy jako dvourozměrné pole i kdyby se jednalo o jeden jediný prvek. Proto se musí pro další použití upravit. Nejprve musíme pomocí funkce **Index Array** získat jednotlivé prvky dvourozměrného pole. Z tohoto důvodu se tato funkce použije dvakrát za sebou (výstup z první je ihned přiveden na výstup druhé). **Index Array** funguje tak, že snižuje řád pole přivedeného na vstup. Proto, když na vstup přivedeme dvourozměrné pole, na výstupu jsou vybraná jednorozměrná pole a při přivedení jednorozměrného pole, jsou už na výstupu prvky tohoto jednorozměrného pole. Dvourozměrné pole datového typu database variants musíme rozdělit na jednotlivé prvky z důvodu toho, že funkce použitá pro převod do datových typů použitelných v LabView **Database Variant to Data** na svůj vstup *Database Variant* umožňuje připojit pouze jednotlivé prvky datového typu database variant. Druhý vstup do této funkce je vstup *type*, na který připojíme konstantu takového datového typu, do jakého chceme datový typ database variant převést. V tomto bloku jsou s touto funkcí použity dvě konstanty. První je řetězcová, použitá pro vytvoření SQL příkazu (získané číslo je převedeno na string pro potřeby dalšího vyhledávání), druhá je číselná, a je použitá pro převod počtu zubů jednotlivých ozubených kol na čísla. Výstup *data* z funkce jsou pak proměnné v datových typech, která už se dají v LabView použít.

Aby načítání z databáze bylo korektně ukončeno, musíme ještě databázi zavřít. Provedeme tak pomocí **DB Tools Close Connection.vi**. Na vstup *connection reference* přivedeme odkaz na otevřenou databázi a virtuální přístroj databázi uzavře.

Výstupem z tohoto bloku (**Načtení počtu zubů**) je jednorozměrné číselné pole array of teeth. To je sestaveno pomocí funkce **Build Array**, z již

zkonvertovaných jednotlivých číselných položek dvourozměrného pole datového typu database variants (výstupy *data* z funkce **Database Variant to Data**). Pole má devět prvků. První tři prvky jsou počty zubů jednotlivých ozubených kol prvního stupně převodovky, další tři druhého stupně a poslední tři třetího stupně. Pokud některá z trojice čísel obsahuje jedničky, převodovka daný stupeň neobsahuje.

4.5 SPOČÍTÁNÍ HLEDANÝCH FREKVENCÍ

- vstupy
    a) array of teeth - jednorozměrné pole počtu zubů jednotlivých ozubených kol načtených z databáze
    b) dt - vzorkovací perioda
- výstupy
    a) wanted frequencies - dvourozměrné pole hledaných frekvencí
    b) strings of wanted frequencies - dvourozměrné pole řetězců hledaných frekvencí

Jelikož průběhy úhlových odchylek (rovnoměrnosti chodu a mrtvého chodu) jsou měřeny na výstupní hřídeli celé převodovky, měla by se frekvence od této výstupní hřídele ve frekvenčním spektru objevit na frekvenci jedna. Z tohoto důvodu se výpočet ostatních frekvencí provádí pomocí druhého sloupce tabulky 3-1. Měřené převodovky nejsou pouze jednostupňové, ale vícestupňové. Vícestupňovou převodovku si lze představit jako více jednostupňových převodovek za sebou. Proto se dá považovat výstupní hřídel předcházejícího stupně za vstupní hřídel následujícího stupně (vstupní hřídel druhého stupně je totožný s výstupní hřídelí prvního stupně). Kolik má převodovka stupňů se dá poznat z jednorozměrného pole počtu zubů array of teeth, které je výstupem z bloku **Načtení počtu zubů**, jak jsem se o tom zmiňoval v předcházející kapitole 4.4. Pokud všechny položky některé trojice z tohoto pole mají hodnotu jedna, převodovka daný stupeň neobsahuje.

Výpočet hledaných frekvencí jsem prováděl v bloku **Formula Node**. Zdál se mně pro tuto činnost nejjednodušší použít kvůli tomu, že se do něho pohodlně zapíší vzorce a přiřazení, podle kterých se frekvence počítají, podobně jako v jazyce C.

Naproti tomu, v blokově orientovaném LabView by tento výpočet, podle mého názoru, byl složitý a nepřehledný (se spoustou bloků pro sčítání, násobení a dělení).

Výpočet postupuje od třetího stupně k prvnímu (důvod je ten, že se úhlové odchylky měří na výstupním hřídeli převodovky, tedy výstupním hřídeli posledního stupně) a probíhá vždy pro každý stupeň zvlášť. Nejprve se pomocí podmínky if zjistí (blok **Formula Node** obsahuje i tyto příkazy jazyka C), jestli převodovka daný stupeň obsahuje (test na nerovnost počtu zubů jedničce). Poté se podle příslušných vzorců kmitočty spočítají. K výpočtu je potřeba vstup právě popisovaného bloku array of teeth (výstup z bloku **Načtení počtu zubů**, který obsahuje počty zubů jednotlivých ozubených kol), který je zároveň vstupem do bloku **Formula Node**. V každém ze vzorců z druhého sloupce tabulky 3-1, podle kterých se počítá, se objevuje kmitočet f_1. Je to kmitočet výstupního hřídele. Ten je pro výpočty posledního stupně nastaven na jedničku, a pro výpočty pro předcházející stupeň na hodnotu kmitočtu vstupního hřídele následujícího stupně (pro tento účel je v bloku **Formula Node** deklarována proměnná fvstup). Vypočítané frekvence jsem skládal do dvourozměrného pole s rozměrem tři krát osm (jsou maximálně tři stupně a každý stupeň obsahuje osm kmitočtů podle tabulky 3-1). Poté jsem ještě provedl korekci spočítaných frekvencí v tomto poli, pokud se v něm objevily frekvence vyšší než je polovina vzorkovací frekvence (tato frekvence je nazývána frekvencí Nyquistovou). Nyquistova frekvence se spočítá v bloku **Formula Node** za pomocí druhého vstupu dt, do právě popisovaného bloku podle známého vzorce:

$$\frac{f_s}{2} = \frac{1}{2 \cdot dt} \qquad \textbf{Rovnice 4-2}$$

kde f_s - vzorkovací frekvence

dt - vzorkovací perioda

Kmitočty vyšší než tato Nyquistova frekvece se potom překlápí ve spektru podle následující rovnice uvedené v[9]:

$$AF = |CIMSF - IF| \qquad \textbf{Rovnice 4-3}$$

[9] National Instruments. Analysis Concepts. March 2004. p.1-6

kde AF - aliased frequency (překlopená frekvence)

CIMSF - closest integer multiple of the sampling freguency (nejbližší celočíselný násobek vzorkovací frekvence)

IF - input frequency (vstupní frekvence)

Korekci jsem provedl také v bloku **Formula Node,** s pomocí dostupných for cyklů, které mají opět stejnou syntaxi jako v jazyce C. Následuje kód, který korekci provádí.

```
for(i = 0; i < 3; i++) {
  for(k = 0; k < 8; k++) {
    if(arrayf[i][k] > fs/2) {
      float32 temp = arrayf[i][k];

      int intdel = 0;
      while(temp > fs) {
        intdel++;
        temp -= fs;
      }
      float32 zbytek = temp%fs;
      if(zbytek > fs/2)
        zbytek = 1;
      else
        zbytek = 0;
      intdel += zbytek;

      arrayf[i][k] = abs(intdel*fs - arrayf[i][k]);
    }
  }
}
```

Tyto dva vnořené for cykly procházejí pole prvek po prvku a testují, jestli není některý z nich větší než Nyquistova frekvence. Pokud tomu tak není, s prvkem se nic neděje. Pokud tomu tak ovšem je, zjistí se nejprve číslo takové, které když vynásobíme vzorkovací frekvencí, dostaneme nejbližší celočíselný násobek vzorkovací frekvence k frekvenci, která byla identifikována jako vyšší než Nyquistova frekvence (toto číslo je nalezeno na řádcích oddělených v zobrazeném

kódu na začátku i na konci prázdným řádkem). Poté už se toto získané číslo použije pro výpočet překlopené frekvence, která následně nahradí v procházeném dvourozměrném poli frekvenci, která je vyšší než Nyquistova frekvence. Výstupem z bloku **Formula Node** je pak toto zkorigované dvourozměrné pole, které je zároveň výstupem wanted frequencies, právě popisovaného bloku. Tyto spočítané, hledané frekvence jsou zobrazeny v tabulce, jak ukazuje obrázek 4.5.

Druhým výstupem z právě popisovaného bloku je strings of wanted frequencies. Je to také dvourozměrné pole s rozměrem tři krát osm. Jsou to řetězce, které budou použity v bloku **Vyhodnocení** a mají následující formát: .fxy, kde x značí stupeň převodovky (1 znamená 1.stupeň) a y značí danou frekvenci (fx2 znamená f2 pro příslušný stupeň). Jsou umístěny tak, aby vždy řetězec v daném formátu, odpovídající dané frekvenci, byl umístěn na stejném místě v poli strings of wanted frequencies, jako spočítaná frekvence v poli wanted frequencies. Toto pole je vytvořeno pomocí funkcí **String Constant** a **Build Array**.

Wanted frequencies

	1.stupeň	2.stupeň	3.stupeň
f0	20,00	0,00	4,00
f1	4,00	0,00	1,00
f2	12,00	0,00	3,00
f3	48,00	0,00	9,00
f4	16,00	0,00	3,00
f5	432,00	0,00	108,00
f6	355,38	0,00	324,00
f7	432,00	0,00	108,00

Obrázek 4.5 Hledané frekvence

4.6 NALEZENÍ A SETŘÍDĚNÍ ŠPIČEK

- vstupy
  - **a)** Amp Spectrum Mag (Vrms) - amplitudové spektrum, získané v bloku **Fourierova transformace**
  - **b)** Amp Spectrum Phase (radians) - fázové spektrum, získané v bloku **Fourierova transformace**
  - c) df - frekvenční rozlišení
  - d) wanted frequencies - hledané frekvence

- výstupy
    a) sequence numbers of peaks - posloupnost pořadí vzorků
    b) frequencies of peaks - frekvence jednotlivých frekvenčních špiček
    c) amplitudes of frequency peaks - amplitudy jednotlivých frekvenčních špiček
    d) phases of frequency peaks - fáze jednotlivých frekvenčních špiček

Tento blok je ve zpracování dat z převodovky velice důležitý. Hledá v nalezeném amplitudovém spektru výrazné frekvenční špičky a přiřazuje k nim fáze z fázového spektra.

K hledání špiček ve spektru jsou použity dva přístupy. První přístup v amplitudovém spektru, za pomoci virtuálního přístroje **Peak Detector.vi** (dále **PD.vi**), vyhledá výrazné harmonické složky. Při samotném použití pouze tohoto přístupu, by mohlo dojít k zanedbání některých důležitých složek frekvenčního spektra, proto jsou harmonické složky vyhledávány ještě jedním přístupem. Využije se k němu dvourozměrného pole hledaných frekvencí (wanted frequecies). Z tohoto pole se berou jednotlivé hledané frekvence a na těchto frekvencích (a jejich násobcích) se přečte hodnota amplitudy v amplitudovém spektru. Poté se pomocí zvoleného algoritmu zkoumá, jestli se tato hodnota dá považovat za špičku nebo nikoli. Potom se sloučí špičky nalezené oběma přístupy do jednoho pole a přiřadí se k nim amplitudy a fáze. Pokud se v tomto poli vyskytuje některá špička dvakrát (tzn. že byla nalezena pomocí obou přístupů), je vždy jedna ze špiček z pole odstraněna.

Jak jsem již popsal v předchozím odstavci, první přístup hledání špiček využívá virtuální přístroj **PD.vi**. Prvním použitým vstupem *X* v tomto virtuálním přístroji, je posloupnost, ve které chceme špičky vyhledat. Do tohoto vstupu připojíme vstup Amp Spectrum Mag (Vrms). Druhým použitým vstupem je vstup *threshold*. Tímto vstupem nastavujeme amplitudu špiček, které mají být v připojené posloupnosti ignorovány (tzn. nejsou již považovány za špičky a **PD.vi** je nevrátí na svém výstupu). Po prozkoumání několika průběhů (z důvodu nastavení této hranice), které jsem měl k dispozici od docenta Beneše a s přihlédnutím k tomu, že se budou špičky hledat ještě jedním způsobem, jsem se rozhodl tuto hranici nastavit jako 15% největší amplitudy v analyzovaném spektru. Toto zajistím pomocí funkce **Array**

Max & Min. Jako její vstup *array* připojím právě analyzované amplitudové spektrum a na výstupu *max value* získám největší hodnotu tohoto spektra. Když ji vynásobím konstantou 0,15, mohu ji přivést na zmiňovaný vstup *threshold*. Posledním vstupem u **PD.vi**, který využívám, je vstup *peaks/valleys*. Tímto vstupem určuji, jestli má **PD.vi** hledat špičky (peaks) a nebo sedla (valleys). Pro moji aplikaci samozřejmě využiji peaks. Prvním využitým výstupem **PD.vi** je *Locations*. Je to posloupnost umístění špiček ve vstupním amplitudovém spektru. Umístěním se myslí pořadové číslo - index špičky ve vstupní posloupnosti. Jelikož tento virtuální přístroj používá tzv. „quadratic fit“ algoritmus k nalezení špiček, který používá interpolaci mezi jednotlivými body, posloupnost umístění nejsou celá čísla. Druhým použitým výstupem je *Amplitudes*. Jak již název napovídá, jsou to amplitudy jednotlivých nalezených špiček. Nyní máme první sadu harmonických složek. Druhou sadu získáme druhým přístupem, který je popsán v podkapitole 4.6.1.

Po získání špiček druhým způsobem máme obě sady a můžeme je sloučit do jednoho pole. Jak jsem již naznačil, může být jedna špička nalezena oběma přístupy. Proto musí být při vzájemném sloučení obou sad špiček, jedna z nich odstraněna, aby ve výsledném poli nebyla dvakrát. Jelikož pomocí prvního přístupu získáme nejen umístění jednotlivých frekvenčních špiček, ale i jejich amplitudy, bude při odstranění ponechána špička získaná právě tímto prvním přístupem. Abych při slučování a odstraňování zdvojených špiček od sebe odlišil špičky získané jedním a druhým způsobem, přiřadil jsem ve smyčce **For Loop** ke špičkám získaným prvním způsobem nulu a ke špičkám získaným druhým způsobem jedničku. Tím pádem mně vznikly dvě jednorozměrná pole clusterů (cluster se skládal ze špičky a přiřazené jedničky nebo nuly), která jsem pomocí funkce **Build Array** sloučil do sebe. Nyní se z tohoto pole odstranily duplikátní špičky pomocí bloku **Remove Duplicates 1D_cluster.vi** (dále jen **RDC.vi**). Tento blok jsem stáhl z internetu[10] a upravil pro použití s clustery. V tomto bloku se nejprve pole, ze kterého bude odstraněna jedna z duplikátních položek, setřídí pomocí funkce **Sorted Array**. Pokud má tato funkce na vstupu *array* pole clusterů a v tomto poli se objeví duplikátní položky na první

[10] http://zone.ni.com/devzone/cda/epd/p/id/2005

pozici v clusteru, třídí se pomocí druhé položky v clusteru. Blok **RDC.vi** poté odstraňuje jednu z duplikátních položek (položky jsou duplikátní na první pozici v clusteru) a to tu, která je v setříděném poli na druhé pozici. Tzn. že odstraní špičku, která má k sobě přiřazenou jedničku, což jsme chtěli. Nyní máme jednorozměrné pole umístění špiček v Amp Spectrum Mag (Vrms).

My ale potřebujeme frekvence špiček, jejich amplitudy a fáze. Tyto náležitosti ke špičkám získáme snadno. Frekvence špiček je vlastně jednorozměrné pole umístění špiček, vynásobené frekvenčním rozlišením df. Amplitudy špiček musíme získat pouze pro špičky získané druhým způsobem, pro první je již máme z výstupu *Amplitudes* z bloku **PD.vi**. Ve smyčce **For Loop** pomocí funkce **Index Array** vybereme ze vstupu Amp Spectrum Mag (Vrms) příslušné amplitudy, pokud smyčku indexujeme jednorozměrným polem špiček, získaných druhým způsobem. Stejně získáme i fáze špiček, musíme však ze vstupu Amp Spectrum Phase (radians) vybrat fáze pro špičky z obou způsobů.

Teď máme tři jednorozměrná pole - pole frekvencí, na kterých se objevily špičky, k nim příslušné amplitudy a fáze. Požadavkem bylo, aby se v tabulce s výsledky zobrazovaly špičky seřazené sestupně podle jejich amplitud. Nestačilo pouze setřídit amplitudy, protože poté už by si amplitudy s fází a frekvencí neodpovídaly. Proto jsem použil jednu vlastnost funkce **Sort 1D Array**. Tato funkce setřídí pole clusterů, podle první položky, pokud takovéto pole přivedeme na vstup *array*. Proto jsem přivedl do smyčky **For Loop** všechna tři jednorozměrná pole (frekvence špiček a jim odpovídající amplitudy a fáze) a s použitím parametru *Enable Indexing* postupně vytvořil jednorozměrné pole clusterů (pomocí funkce **Bundle**). Každý cluster v poli se nyní skládá ze tří prvků *double*, z nichž první je amplituda. Na takto vytvořené pole jsem aplikoval již zmiňovanou funkci **Sort 1D Array**, která mi toto pole setřídila od nejmenšího po největší. Chtěl jsem to přesně naopak, proto jsem ještě použil funkci **Reverse 1D Array.** Nyní jsem toto pole přeměnil zpět z pole clusterů na tři obyčejná jednorozměrná pole (pomocí **Unbundle**). Tato pole jsou již výstupy z právě popisovaného bloku (frequencies of peaks, amplitudes of frequency peaks, phases of frequency peaks). Čtvrtý výstup sequence numbers of peaks se získá při výběru fází k příslušným špičkám ve smyčce

For Loop. Jedná se o jednorozměrné pole, které se složí z *iteration* terminálu zvětšeného o jedničku.

4.6.1 Špičky na spočítaných frekvencích

- vstupy
  - **a)** Amp Spectrum Mag (Vrms) - amplitudové spektrum, získané v bloku **Fourierova transformace**
  - b) df - frekvenční rozlišení
  - c) wanted frequencies - hledané frekvence
- výstupy
  - a) peaks - špičky získané tímto virtuálním přístrojem

Druhou sadu špiček dostaneme pomocí bloku **Špičky na spočtených f**. Tento blok hledá velikosti amplitud v amplitudovém spektru na hledaných frekvencích (vstup wanted frequencies), až do jejich patnáctých harmonických (pokud harmonické od hledané frekvence nepřekročí frekvenční rozsah) a určuje, jestli se dá tato hodnota považovat za špičku, nebo ne.

Nejprve musíme z dvourozměrného pole hledaných frekvencí (vstup wanted frequencies) s rozměrem tři krát osm odstranit jednorozměrná pole, obsahující samé nuly. Tato jednorozměrná pole jsou známkou toho, že převodovka neobsahuje některý stupeň. Odstranění se provede následujícím způsobem. Je využito smyčky **For Loop**, kterou indexuje dvourozměrné pole wanted frequencies (je povoleno indexování smyčky, položka *enable indexing*). Tzn. že se smyčka provede třikrát. Je to z toho důvodu, že pole obsahuje tři jednorozměrná pole. Dále jsou ve smyčce dva *shift* registry. Ty jsou výhodné v tom, že si pamatují hodnotu z předchozího cyklu. Tyto registry jsou inicializovány, jako dvourozměrná pole obsahující samé nuly pomocí funkce **Initialize Array**. V každém cyklu for smyčky se nejprve zjistí, jestli aktuální jednorozměrné pole obsahuje nuly. Pokud ano, je připojeno k dvourozměrnému poli nul, pokud ne, je připojeno k nenulovému dvourozměrnému poli. K tomu se použije blok **Case Structure**. Z jednorozměrného pole se pomocí funkce **Index Array** vybere jeden prvek a vede se do funkce **Equal To 0?**. Ta na svém výstupu dává logickou hodnotu true nebo false, která je dále přivedena na vstup *case selector* bloku **Case Structure**. Podle této hodnoty se dané

jednorozměrné pole připojí k nulovému nebo nenulovému dvourozměrnému poli v **Case Structure** pomocí funkce **Build Array**. Dále se pracuje už pouze s nenulovým dvourozměrným polem, které je na výstupu for smyčky vyvedeno z jednoho ze *shift* registrů.

Dále si musíme vytvořit posloupnost frekvenčních vzorků (vodorovná, x-ová osa v amplitudovém spektru). Tato posloupnost se vytvoří pomocí **For Loop** s jedním *shift* registrem a pomocí vstupu df (frekvenčního rozlišení). *Shift* registr se inicializuje hodnotou 2,5E-5 (což je hodnota 0,000025). Toto je první hodnota pole frekvenčního rozlišení. Potom se v prvním cyklu k tomuto číslu přičte df a v každém následujícím cyklu hodnota z předcházejícího cyklu plus df. Tyto hodnoty se pak na výstupu ze smyčky postupně skládají do jednorozměrného pole (položka *enable indexing*).

Samotný algoritmus vyhledávání špiček probíhá ve smyčce **While Loop**, která je uvnitř dvou do sebe vnořených smyček **For Loop** (smyčky **For Loop** jsou zde proto, aby se prošlo celé upravené dvourozměrné pole wanted frequencies). Smyčka **While Loop** je provedena maximálně patnáctkrát (Hledá se do patnácté harmonické, jak bylo uvedeno na začátku této kapitoly.). Může být také provedena méněkrát, pokud některý násobek frekvence, na které se právě hledá, překročí frekvenční rozsah (Nyquistovu frekvenci). Tyto dvě podmínky jsou sestaveny tak (pomocí knihoven **Comparison** a **Boolean**), aby při splnění alespoň jedné z nich, došlo k ukončení **While Loop**. To, jestli se na dané frekvenci (nebo jejím násobku) nachází špička nebo ne, je určeno v následně popsané části programu.

Nejprve se vybere vzorek z posloupnosti frekvenčních vzorků, který se nejvíce blíží svou hodnotou k hledané frekvenci. Tzn., že se vydělí hledaná frekvence (nebo její harmonická) frekvenčním rozlišením df a poté se pomocí funkce **Index Array** vybere (celá část dělení)-tý vzorek z posloupnosti frekvenčních vzorků a vzorek o jedno větší. Hodnota indexu s bližší hodnotou k dané frekvenci projde dál přes **Case Structure**. V **Case Structure** je ještě vybrána hodnota příslušné amplitudy k dané frekvenci z jednorozměrného pole a zároveň vstupu Amp Spectrum Mag (Vrms) (pomocí funkce **Index Array**). Nyní se určí, jestli se dá amplituda

na hledané frekvenci považovat za špičku nebo ne. Zvolil jsem pro to následující kritérium.

Nejprve se vybere deset amplitud před a po hledané frekvenci. Poté se v této posloupnosti (má 21 položek - amplituda na hledané frekvenci plus deset položek před a deset položek po) hledají špičky (pomocí již známého virtuálního přístroje **Peak Detector.vi**) s tím, že se zanedbávají špičky menší než polovina hodnoty amplitudy na hledané frekvenci (vstup *threshold*). Potom, pokud je špička pouze jedna a je na hledané frekvenci, dá se amplituda považovat za špičku a číslo vzorku projde přes case structure. Pokud toto splněno není, projde nula. To, že je špička pouze jedna, se zjistí pomocí funkce **Array Size**. Té, pokud přivedeme na vstup *array* výstup *Locations* z **Peak Detectoru.vi,** musí vrátit jedničku. A dále, že je špička na hledané frekvenci, se zjistí pomocí funkce **Index Array**. Ta, pokud bude mít na vstupu *array* zaokrouhlený výstup *Locations* (pomocí funkce **Round To Nearest**), musí mít první prvek výstupu *element or subarray* roven deseti. Tzn. jedenáctý prvek v posloupnosti, což je právě hledaná frekvence nebo její násobek.

Špičky, získané tímto způsobem, jsou nyní ve formě dvourozměrného pole (je výstupem z dvou do sebe vnořených **For Loop**), které ale může obsahovat nulové položky, pokud na dané frekvenci nebo jejím násobku nebyla identifikována špička. Proto je toto pole pomocí funkce **Reshape Array** převedeno na jednorozměrné pole a jsou z něj odstraněny nulové položky, což je provedeno stejným způsobem, již jednou popsaným v této podkapitole. Toto jednorozměrné pole je již výstupem peaks z tohoto bloku.

4.7 VYHODNOCENÍ

- vstupy
    a) strings of wanted frequencies - dvourozměrné pole řetězců hledaných frekvencí
    b) wanted frequencies - hledané frekvence
    c) frequencies of peaks - frekvence nalezených špiček
- výstupy

a) array of notes - jednorozměrné pole poznámek pro zobrazení výsledků v tabulce

Tento blok zjišťuje, zda jsou nalezené špičky z bloku **Nalezení a setřídění špiček** shodné s hledanými frekvencemi a nebo jejich násobky. Shodnost neznamená, že se frekvence nalezené špičky přesně shoduje s hledanou frekvencí, ale znamená, že je nalezená špička v malém rozmezí okolo hledané frekvence (nebo jejího násobku). Toto malé rozmezí se určuje pro každou frekvenci zvlášť v bloku **Hystereze**, který bude popsán v podkapitole 4.7.1. Pokud je tedy špička shodná s hledanou frekvencí, jsou na základě této shodnosti ze vstupu strings of wanted frequencies vytvořeny poznámky, které jsou zobrazeny v tabulce jako notes.

Test na shodnost probíhá ve třech smyčkách **For Loop**, které jsou vnořeny do sebe. Smyčka, která je nejvíce vnější, proběhne tolikrát, kolik je nalezených špiček ve frekvenčním spektru, protože vstup frequencies of peaks tuto smyčku indexuje. Další dvě vnitřní smyčky jsou indexovány od dalších dvou vstupů strings of wanted frequencies a wanted frequencies (jedná se o dvourozměrná pole, která mají stejné rozměry). V nejvnitřnější smyčce je nejprve vydělena nalezená špička hledanou frekvencí pomocí funkce **Quotient & Remainder**. Na vstupy této funkce se přivede dělenec a dělitel a na výstupu *x-y*floor(x/y)* dostaneme zbytek po dělení, zatímco na výstupu *floor(x/y)* dostaneme výsledek celočíselného dělení. Dalšími prvky v nejvnitřnější smyčce jsou dva bloky **Case Structure** (dále **CS**) vnořené do sebe. Do *case selectoru* vnějšího **CS** je přiveden výsledek logické operace **Greater Or Equal?** (větší nebo rovno). Tato operace je zde z toho důvodu, že nalezená špička může být maličko vychýlena vpravo nebo vlevo od hledané frekvence (nebo jejího násobku). Proto pokud je zbytek po celočíselném dělení (výstup *x-y*floor(x/y)*) větší než polovina hledané frekvence, nachází se nalezená špička spíše vlevo od případné hledané frekvence. Naopak pokud je zbytek menší, nachází se nalezená špička spíše vpravo od případné hledané frekvence. Do vnějšího **CS** jsou jako vstupy přivedeny hledaná frekvence, její string (řetězec jejího popisu ve formátu .fxy, který byl popsán v kapitole 4.5), výsledek zmiňovaného celočíselného dělení a zbytek po celočíselném dělení (dělenec je nalezená špička, dělitel hledaná frekvence).

Případ True vnějšího **CS** je případ, kdy je potenciálně shodná špička spíše vlevo od hledané frekvence. Proto se testuje rozdíl hledané frekvence a zbytku po celočíselném dělení, jestli je v přijatelné mezi. To by znamenalo shodnost špičky s hledanou frekvencí. Tuto mez určuje výstup *left_hysteresis* bloku **Hystereze**. Pokud je tedy rozdíl hledané frekvence a zbytku po celočíselném dělení menší než výstup *left_hysteresis*, je nalezená špička shodná s hledanou frekvencí nebo jejím násobkem. Test se provádí funkcí **Less Or Equal?** jejíž výstup (logická hodnota) je přiveden do *case selectoru* vnitřního **CS**. V tomto **CS** se již v případu (case) True tvoří poznámky, které budou použity v tabulce.

Případ False vnějšího **CS** je případ, kdy je potenciálně shodná špička spíše vpravo od hledané frekvence. Zde se testuje zbytek po celočíselném dělení na přijatelnou mez. Tentokrát je mez určena jako výstup *right_hysteresis* bloku **Hystereze**. Pokud je tedy zbytek po celočíselném dělení menší než výstup *right_hysteresis*, je nalezená špička shodná s hledanou frekvencí nebo jejím násobkem. Test se provádí stejným způsobem, jako v případě True a jeho výsledek je také přiveden do *case selectoru* vnitřního **CS**.

Ve vnitřním **CS** se tvoří případná poznámka použitá v tabulce. Tato poznámka je ve tvaru „z.fxy“. Formát .fxy je popsaný v kapitole 4.5, číslo „z“ je výsledek celočíselného dělení. Tento výsledek nám vlastně určuje kolikátá harmonická, od hledané frekvence, je nalezená špička. Tento výsledek musí být, pro případ True vnějšího **CS,** ještě zvětšen od jedničku. Poznámka tedy vznikne sloučením výsledku celočíselného dělení a stringu .fxy (ze vstupu strings of wanted frequencies) pomocí funkce **Concatenate Strings**. Předtím však musí být výsledek celočíselného dělení převeden z čísla na string. To se provede pomocí funkce **Number To Decimal String**. Až proběhnou všechny cykly nejvnitřnější smyčky, obsahuje prostřední smyčka jednorozměrné pole stringů poznámek k právě analyzované špičce. Toto pole se sloučí, pomocí funkce **Concatenate Strings,** do jednoho stringu. To samé se provede i u nadřazené smyčky. Výstupem, z nejvíce vnější smyčky, je potom jednorozměrné pole stringů, které má stejnou délku jako pole nalezených frekvenčních špiček. Toto pole je výstupem array of notes z právě popisovaného bloku. Každý prvek tohoto pole je string, který tvoří poznámku

k nalezené špičce o její shodnosti s hledanými frekvencemi nebo jejími násobky. Jednotlivé poznámky pro jednu špičku jsou odděleny mezerami.

4.7.1 Hystereze

- vstupy
    a) frequency - frekvence, pro kterou se má hystereze určovat
- výstupy
    a) left_hysteresis - maximální možná výchylka špičky vlevo, pokud má být ještě považována za shodnou s hledanou frekvencí
    b) right_hysteresis - maximální možná výchylka špičky vpravo, pokud má být ještě považována za shodnou s hledanou frekvencí

Tento blok nám určuje maximální možnou odchylku nalezené špičky od hledané frekvence nebo jejího násobku, abychom mohli ještě nalezenou špičku považovat za shodnou s hledanou frekvencí. Využijeme toho, že známe vzorkovací frekvenci analyzovaných průběhů rovnoměrnosti chodu a mrtvého chodu a délku jejich záznamu. Proto můžeme pro každou hledanou frekvenci zvlášť spočítat amplitudové spektrum a z něho určit tuto maximální odchylku.

Vstupem do tohoto bloku je vstup frequency. Tento vstup přivedeme na vstup *frequency* virtuálního přístroje **Sine Waveform.vi**, pomocí něhož je vytvořen sinusový signál. Dalším použitým vstupem tohoto virtuálního přístroje je vstup *sampling info*, kam jsou přivedeny informace o vzorkovací frekvenci a počtu vzorků, aby se přesně shodovaly podmínky vytvořeného signálu a záznamu průběhu rovnoměrnosti chodu nebo mrtvého chodu. Na výstupu *signal out* **Sine Waveform.vi** dostaneme příslušný signál. Z tohoto signálu je spočítáno amplitudové spektrum, pomocí **Amplitude and Phase Spectrum.vi**. Jelikož tento virtuální přístroj potřebuje na svých vstupech vzorkovací periodu a hodnoty amplitudy příslušného signálu, musíme tato data získat pomocí funkce **Get Waveform Components** z výstupu *signal out*. Tento výstup je totiž datového typu „waveform“. Z výstupu *Amp Spectrum Mag* (z virtuálního přístroje **Amplitude and Phase Spectrum.vi**) získáme amplitudové spektrum signálu, které je prvním vstupem do bloku **Formula Node**. V tomto bloku následně probíhá počítání hystereze. Dalším vstupem do bloku **Formula Node** je vodorovná osa amplitudového spektra, což jsou zlomky frekvence,

na kterých jsou příslušné amplitudy vytvořeného signálu. Tato posloupnost frekvenčních vzorků se vytvoří stejným způsobem jako bylo popsáno v kapitole 4.6.1 pomocí **For Loop** a jednoho *shift* registru. Posledním vstupem do **Formula Node** je hodnota amplitudy největšího vzorku v amplitudovém spektru a jeho index. Tyto dvě položky se získají pomocí funkce **Array Max & Min**.

Hysterezi (neboli maximální odchylku nalezené frekvence od hledané, kdy je ještě možné považovat nalezenou špičku za shodnou s hledanou frekvencí) jsem uvažoval jako pokles o 3dB vlevo i vpravo od maxima spočítaného amplitudového spektra. Pokles o 3dB se dají vyjádřit jako toto maximum násobené konstantou $\frac{\sqrt{2}}{2}$. Hystereze je tedy rozdíl frekvence, na které je amplituda vpravo od maxima ve spočítaném amplitudovém frekvenčním spektru $\frac{\sqrt{2}}{2}$ krát menší než toto maximum a frekvence, na které je amplituda vlevo od maxima ve spočítaném amplitudovém frekvenčním spektru $\frac{\sqrt{2}}{2}$ krát menší než toto maximum. Jelikož máme amplitudové spektrum diskrétní, nemáme zaručeno, že se nám v tomto spektru objeví vzorek s amplitudou přesně $\frac{\sqrt{2}}{2}$ krát menší, než je maximum tohoto spektra. Proto jsem zvolil přístup, že procházím postupně vzorky napravo i nalevo od maxima spočítaného spektra a určím si vždy dva vzorky pro každou stranu, z nichž jeden je se svou amplitudou nad hranicí poklesu o 3dB a druhý je pod touto hranicí. Mezi těmito dvěma vzorky proložím přímku a jelikož znám obě souřadnice těchto dvou bodů, mohu určit přesně hodnotu frekvence, kde je pokles amplitudy o 3dB oproti maximu amplitudového frekvenčního spektra. Jelikož to dělám pro každou stranu zvlášť, jsou výstupy z tohoto bloku dva a to *left_hysteresis* a *right_hysteresis*. Uvažuji rovnici přímky:

$$y = kx + q$$ **Rovnice 4-4**

Pokud mám dva body (o souřadnicích $[x_1,y_1]$ a $[x_2,y_2]$) mohu určit koeficienty k a q pomocí rovnic

$$k = \frac{y_1 - y_2}{x_1 - x_2}$$ **Rovnice 4-5**

$$q = y_2 - kx_2$$ **Rovnice 4-6**

Potom frekvenci, na které je právě pokles o 3 dB, spočítám pomocí rovnice

$$f_{3dB} = \frac{mez - q}{k}$$ **Rovnice 4-7**

kde mez je hodnota maxima amplitudového spektra násobená konstantou $\frac{\sqrt{2}}{2}$.

Hodnota výstupu *left_hysteresis* je potom rozdíl frekvence maxima amplitudového spektra a hodnoty f_{3dB} vlevo od maxima. Hodnota *right_hysteresis* je rozdíl hodnoty f_{3dB} napravo od maxima a frekvence maxima amplitudového spektra. Nalezení příslušných dvojic vzorků, proložení přímek a výpočet frekvence levé a pravé hystereze se provádí v bloku **Formula Node**.

4.8 ZOBRAZENÍ

- vstupy
    a) sequence numbers of peaks - posloupnost pořadí vzorků
    b) frequencies of peaks - frekvence jednotlivých frekvenčních špiček
    c) amplitudes of frequency peaks - amplitudy jednotlivých frekvenčních špiček
    d) phases of frequency peaks - fáze jednotlivých frekvenčních špiček
    e) array of notes - jednorozměrné pole poznámek k frekvenčním špičkám
- výstupy
    c) 2D array of strings to table - dvourozměrné pole řetězců pro zobrazení v tabulce

Tento blok slouží ke konverzi získaných číselných dat na řetězce, aby bylo možné je přehledně zobrazit v tabulce. Každý vstup je přiveden do funkce **Number To Fractional String**, která každý ze vstupů převede na jednorozměrné pole stringů. U této funkce byl ještě použit vstup *precision*, který určuje počet desetinných míst

v řetězci. Pro vstup sequence numbers of peaks je použita konstanta nula (není potřeba desetinná místa), pro ostatní vstupy konstanta čtyři. Poté se tato jednorozměrná pole sloučili do jednoho dvourozměrného pole pomocí funkce **Build Array** a ještě se toto pole pomocí funkce **Transpose 2D Array** transponovalo, aby byli všechny informace k nalezené špičce v jednom řádku.

Výsledná tabulka, která zobrazuje výstupní dvourozměrné pole obsahuje pořadové číslo vzorku, frekvenci (v mém případě v jednotkách 1/ot), příslušnou amplitudu a fázi a nakonec poznámku (notes). Ta značí, od kterých ozubených kol by mohla příslušná frekvenční složka být. Ukázka tabulky je na obrázku 4.6.

Table - uniformity right

number	1/ot	amplitude	phase[rad]	notes
1	18,9547	0,2398	29,2851	19.f31
2	2,0135	0,1384	11,9964	2.f31
3	15,9619	0,1310	20,8164	4.f11 1.f14 4.f30 16.f31
4	1,0121	0,1263	9,5463	1.f31
5	3,0394	0,1234	10,9425	3.f31 1.f32 1.f34
6	52,0125	0,1168	8,1371	13.f11 13.f30 52.f31
7	17,9308	0,1147	23,2672	2.f33
8	19,9836	0,0886	29,0886	1.f10 5.f11 5.f30 20.f31
9	21,9794	0,0820	31,2033	22.f31
10	9,0436	0,0765	24,5849	9.f31 3.f32 1.f33 3.f34
11	4,0132	0,0729	12,2146	1.f11 1.f30 4.f31
12	38,0176	0,0632	14,5330	38.f31
13	29,0163	0,0589	35,9499	29.f31
14	17,1467	0,0583	23,5923	
15	22,9898	0,0571	35,6494	23.f31
16	8,0176	0,0543	25,0692	2.f11 2.f30 8.f31

Obrázek 4.6 Tabulka se zobrazenými výsledky analýzy

4.9 PRAVDĚPODOBNOST

- vstupy
    a) array of notes - poznámky k nalezeným frekvenčním špičkám
    b) amplitudes of frequency peaks - amplitudy nalezených frekvenčních špiček
- výstupy
    a) probability array - dvourozměrné pole řetězců pro přibližné určení pravděpodobnosti

b) 2D array of distribution amplitudes along frequencies - dvourozměrné pole čísel pro přibližné určení pravděpodobnosti

Tento blok z jednorozměrného pole poznámek (array of notes) a jednorozměrného pole amplitud nalezených frekvenčních špiček vytváří dvourozměrné pole řetězců, které je poté zobrazeno v tabulce a grafu. Každému sloupci v tomto poli přísluší jedna z nalezených frekvencí (ve formátu fxy). Řádky tvoří amplitudy harmonických k těmto nalezeným frekvencím až do patnácté harmonické.

Pro vytvoření tohoto pole potřebujeme nejprve jednorozměrné pole řetězců hledaných frekvencí (ve formátu fxy), ke kterým byla nalezena alespoň jedna harmonická složka. Proto musíme projít vstup array of notes prvek po prvku a z každého tohoto prvku vybrat pouze tuto část řetězce a zajistit, aby vybrané části řetězců nalezených frekvencí byly v tomto poli unikátní. Zajistíme to pomocí smyčky **For Loop**, kterou nám bude indexovat vstup array of notes. V této smyčce je *shift* registr, s jehož pomocí se bude postupně skládat požadované pole řetězců. Jelikož může poznámka obsahovat více částí ve formátu z.fxy oddělených mezerami, je ve for smyčce ještě smyčka **While Loop**. V této smyčce se pomocí funkce **Match Pattern** vyhledává v řetězci připojeném na vstup *string* část řetězce, která má formát připojený na vstup *regular expression* (v našem případě f[0-9]+, protože vyhledáváme část řetězce ve formátu fxy, kde xy je číslo). Na výstupu *match substring* dostanu nalezenou část řetězce, pokud ho řetězec na vstupu obsahuje. Dalším využitým výstupem je *offset past match*, který je připojen do *shift* registru while smyčky. Z tohoto výstupu dostanu index znaku následujícího za částí řetězce, kterou jsem hledal, pokud je tato část nalezena. Pokud nalezena není, je vrácena mínus jednička. Tento výstup je použit pro ukončení smyčky **While Loop** (smyčka se ukončí, pokud je roven mínus jedničce) a také jako vstup *offset* funkce **Match Pattern** v následujícím cyklu while smyčky (do tohoto vstupu je totiž připojen *shift* registr, který je pro první cyklus while smyčky inicializován na nulu). Vstupem *offset* určujeme od jakého indexu má funkce **Match Pattern** začít vyhledávat příslušnou část řetězce. Z nalezených částí ještě odstraníme písmeno f (formát fxy přejde na xy), protože budeme nalezené části poznámek převádět na čísla, abychom z nich mohli

odstranit duplikátní položky. Písmeno f bude pro zobrazení v tabulce opět dodáno. Odstranění provedeme pomocí funkce **String Subset**. Na výstupu while smyčky je po každém cyklu jednorozměrné pole stringů ve formátu „xy“, které postupně připojujeme, za pomoci funkce **Build Array**, k poli získanému v předcházejících cyklech for smyčky pomocí jejího *shift* registru. Na výstupu z for smyčky je toto sloučené jednorozměrné pole, které pomocí funkce **Decimal String To Number** převedu na číselné pole. Toto pole přivedu na vstup **Remove Duplicates 1D_array.vi**, které mi odstraní duplikátní položky. Tento virtuální přístroj jsem stáhl z internetu, jak je uvedeno v kapitole 4.6. Teď si vytvořím dvourozměrné pole pravděpodobnosti (prozatím samých nul) pomocí funkce **Initialize Array**. Jeho rozměry jsou patnáct (budou se skládat amplitudy do patnácté harmonické) krát počet unikátních nalezených frekvencí, získaných na výstupu virtuálního přístroje **Remove Duplicates 1D_array.vi**. Do tohoto pole nyní budeme vpisovat jednotlivé amplitudy.

Vpisování probíhá ve smyčce **For Loop** uvnitř které je opět smyčka **While Loop**. Vnější for smyčka je indexována od vstupů array of notes a amplitudes of frequency peaks (obě pole mají stejné rozměry). Dalšími vstupy do for smyčky jsou jednorozměrné pole unikátních nalezených frekvencí (ve formátu xy, kde xy je číslo) a inicializované dvourozměrné pole samých nul, které je připojeno do *shift* registru for smyčky (musí zůstat zachovány změny ze všech cyklů). Tyto čtyři vstupy jsou také zároveň vstupy while smyčky s tím, že dvourozměrné pole samých nul je připojeno do *shift* registru while smyčky. V této smyčce se zpracovává vždy jedna poznámka. Tentokrát se v poznámce nehledá část řetězce ve formátu „fxy“, ale celé části oddělené mezerou ve formátu „z.fxy“. Budeme totiž potřebovat nejen číslo „xy“, abychom věděli do kterého sloupce máme amplitudu zapsat, ale i číslo „z“, abychom věděli do kterého řádku číslo zapíšeme. Vyhledáváme opět pomocí **Match Pattern**, ale na vstup *regular expresssion* je nyní připojena konstanta [0-9]+.f[0-9]+. Výstup *offset past match* je použit stejně, jako v předchozím vyhledávání (pro ukončení while smyčky a jako vstup *offset* **Match Pattern** pro následující cyklus). Výstup *match substring* je přiveden na vstup *string* dalšího **Match Pattern**, abychom od sebe oddělili číslo „z“ a „xy“ z formátu „z.fxy“. Použijeme k tomu ještě

funkci **String Subset** a obě části získaného řetězce převedeme pomocí **Decimal String To Number** na čísla. Samotné vložení amplitudy do dvourozměrného pole pravděpodobnosti je vykonáno v bloku **Formula Node**. Při shodě čísel ve formátu „xy“ z pole unikátních nalezených frekvencí a čísla „xy“, které bylo separováno z části řetězce „z.fxy“, se zapíše aktuální hodnota amplitudy do „z“-tého řádku (číslo „z“ určuje, kolikátá amplituda od hledané frekvence je daná špička) a „xy“-tého sloupce dvourozměrného pole pravděpodobnosti.

Toto dvourozměrné pole pravděpodobnosti (je výstupem for smyčky) musí být převedeno na řetězce (stringy), aby mohlo být zobrazeno v tabulce. Provede se to pomocí dvou **For Loop** vnořených do sebe. Ve vniřní for smyčce je **Case Structure**, do jehož *case selectoru* se přivede výsledek z **Equal?**. Na vstup této funkce připojím každý jednotlivý prvek dvourozměrného pole pravděpodobnosti a nulu. Poté případ True (tzn. že prvek je nulový) převede číselný prvek na řetězec pomocí funkce **Number To Fractional String,** se vstupem *precision* nastaveným na nulu. Případ False (prvek není roven nule) převede číslo na řetězec také pomocí **Number To Fractional String**, nyní však se vstupem *precision*, nastaveným na čtyři. Je to z důvodu, že takto bude zobrazení tohoto pole v tabulce přehlednější. Nenulové hodnoty amplitud mají čtyři desetinná místa, zatímco nuly jsou zobrazeny čistě jako nuly. K tomuto dvourozměrnému poli stringů je ještě připojeno jednorozměrné pole stringů unikátních nalezených frekvencí ve formátu „xy“, doplněného ve smyčce **For Loop** a pomocí funkcí **Number To Decimal String** a **Concatenate Strings** o „f“. Připojené pole má tedy jednotlivé prvky ve formátu „fxy“. Toto pole (dvourozměrné pole pravděpodobnosti plus jednorozměrné pole s prvky ve formátu „fxy“) je výstupem probability array a bude zobrazeno v tabulce. Druhým výstupem je 2D array of distribution amplitudes along frequencies, což je dvourozměrné číselné pole pravděpodobnosti před konverzí na stringy. Tento výstup bude zobrazen v tzv. „intensity“ grafu.

4.10 ZOBRAZENÍ VÝSLEDKŮ PRAVDĚPODOBNOSTI

Zobrazení výsledků pravděpodobnosti se skládá ze zobrazení výstupů z bloku **Pravděpodobnost** probability array v tabulce a 2D array of distribution amplitudes

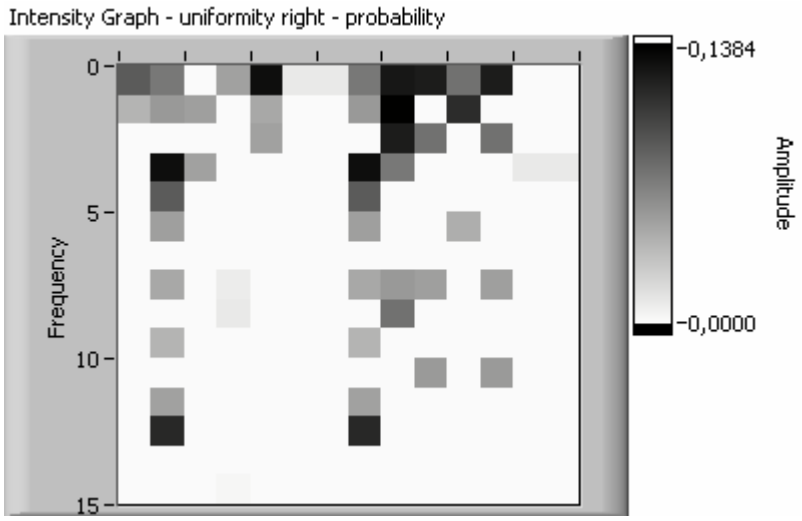
along frequencies v „intesity“ grafu. Ukázka tabulky je na obrázku 4.7. a ukázka grafu je na obrázku 4.8.

Zobrazení v „intensity“ grafu, je zobrazení se třemi stupni volnosti. Máme dvě osy a navíc třetí rozměr, což je intenzita zobrazovaného bodu. Na x-ové ose jsou unikátní nalezené frekvence, na y-ové ose potom jednotlivé harmonické (od první do patnácté). Třetí rozměr (intenzita zobrazovaného bodu) je amplituda. Zobrazení amplitudy jsem si nastavil tak, pomocí **Property Node** „intensity“ grafu (položka ZScale.MarkerVals[]), že nulová amplituda je zobrazena jako bílá a maximální amplituda jako černá.

Probability - uniformity right

	f10	f11	f12	f13	f14	f15	f17	f30	f31	f32	f33	f34	f35
1.	0,0886	0,0729	0	0,0497	0,1310	0,0100	0,0100	0,0729	0,1263	0,1234	0,0765	0,1234	0
2.	0,0394	0,0543	0,0507	0	0,0461	0	0	0,0543	0,1384	0	0,1147	0	0
3.	0	0	0	0	0,0497	0	0	0	0,1234	0,0765	0	0,0765	0
4.	0	0,1310	0,0497	0	0	0	0	0,1310	0,0729	0	0	0	0,0100
5.	0	0,0886	0	0	0	0	0	0,0886	0	0	0	0	0
6.	0	0,0507	0	0	0	0	0	0,0507	0	0	0,0425	0	0
7.	0	0	0	0	0	0	0	0	0	0	0	0	0
8.	0	0,0461	0	0,0084	0	0	0	0,0461	0,0543	0,0507	0	0,0507	0
9.	0	0	0	0,0100	0	0	0	0	0,0765	0	0	0	0
10.	0	0,0394	0	0	0	0	0	0,0394	0	0	0	0	0
11.	0	0	0	0	0	0	0	0	0	0,0534	0	0,0534	0
12.	0	0,0497	0	0	0	0	0	0,0497	0	0	0	0	0
13.	0	0,1168	0	0	0	0	0	0,1168	0	0	0	0	0
14.	0	0	0	0	0	0	0	0	0	0	0	0	0
15.	0	0	0	0,0028	0	0	0	0	0	0	0	0	0

Obrázek 4.7 Zobrazení dvourozměrného pole pravděpodobnosti v tabulce



Obrázek 4.8 Zobrazení dvourozměrného pole pravděpodobnosti v grafu

4.11 VZHLED ČELNÍHO PANELU APLIKACE

Čelní panel aplikace je koncipován do tzv. **Tab control**. Je to plocha, do které jsou umístěny všechny ostatní ovladače a indikátory. Celý **Tab control** je rozdělen na šest záložek, z nichž tři a tři jsou stejné.

První tři záložky obsahují výsledky analýzy průběhů rovnoměrnosti chodu vlevo, vpravo a mrtvý chod. Je na nich umístěn vždy průběh rovnoměrnosti chodu nebo mrtvého chodu, jeho amplitudové frekvenční spektrum, tabulka s hledanými frekvencemi pro všechny stupně a tabulka s výsledky analýzy. Je zde také uveden popis jednotlivých hledaných frekvencí, aby se v nich dokázal uživatel orientovat. Ukázka jedné této záložky je uvedena v příloze 1.

Druhé tři záložky obsahují výsledky přibližného určení pravděpodobnosti a kepstrální analýzy rovněž pro všechny tři analyzované průběhy. Těmito výsledky se rozumí tabulka z bloku **Pravděpodobnost** a zobrazení dat z tabulky v Intensity grafu. Je připojen ještě popis co tabulka a graf představují, aby byl uživatel v obraze. Výsledek kepstrální analýzy je zobrazen v grafu a nemá vypovídací hodnotu. Nejsou z něj totiž vyvozovány žádné závěry. Ukázka jedné z těchto tří záložek je v příloze 2.

5. ZÁVĚR

Vytvořený modul analyzuje naměřené průběhy rovnoměrnosti chodu a mrtvého chodu z planetových převodovek. Naměřené průběhy z harmonických převodovek analyzovat neumí, protože jsem je neměl k dispozici a neměl jsem k dispozici ani databázi harmonických převodovek. Nevidím v tomto nedostatku závažný problém, protože úprava modulu za účelem analýzy průběhů i z harmonických převodovek by nestála mnoho času a sil. Upravit by se musel pouze blok **Spočítání hledaných frekvencí**, což je, při znalosti rovnice pro převodový poměr, banalita. Pokud by byli počty zubů jednotlivých komponent harmonické převodovky jinak uspořádány v databázi než počty zubů převodovek planetových, musel by se upravit i blok **Načtení počtu zubů**. Úpravy by však nebyly nijak rozsáhlé.

Modul nejprve načte naměřená data ze souboru, který vybereme a poté je převede do frekvenční oblasti. Z cesty k aktuálně analyzovanému souboru vypreparuje číslo pomocí něhož načte zuby jednotlivých ozubených kol planetové převodovky z databáze. Tento blok je velice důležitý, protože značně zrychluje a zjednodušuje analýzu. Pokud by bylo k dispozici pouze „ruční“ načítání z databáze, analýza většího počtu průběhů by byla téměř nemyslitelná. Pomocí nalezených počtů zubů jsou spočítány hledané frekvence. V datech, převedených do frekvenční oblasti, jsou vyhledány výrazné frekvenční špičky a setříděné podle velikosti amplitud těchto špiček. Špičky jsou dále testovány na shodnost s hledanými frekvencemi. Pokud ke shodnosti dochází, jsou vytvořeny poznámky k jednotlivým špičkám. Poté jsou hodnoty frekvence, amplitud, fází a poznámek k jednotlivým špičkám zobrazeny v tabulce. Je také vytvořen blok, pro přibližné stanovení pravděpodobnosti přiřazení získaných špiček jednotlivým ozubeným kolům.

Velkým pomocníkem při tvorbě tohoto modulu mně byla nápověda a příklady v LabView. S jejich pomocí jsem vyřešil spoustu problémů velmi rychle a jednoduše. V neposlední řadě mě pomohl i internet, kde jsem našel jeden virtuální přístroj, který používám ve své aplikaci a spoustu dalších informací.

6. LITERATURA

[1] Uhlíř, J., Sovka, P., Číslicové zpracování signálů, Vydavatelství ČVUT, Praha 1995

[2] National Instruments. Analysis Concepts. March 2004 - zdroj http://www.ni.com/pdf/manuals/370192c.pdf

[3] Kreidl, M., Šmíd, R., Technická diagnostika - senzory, metody, analýza signálu, BEN, 2006, ISBN 80-7300-158-6

[4] Malec, Z., Beneš, P., Klusáček, S. Vývoj metod pro měření parametrů přesných převodovek : závěrečná zpráva projektu GA ČR 102/01/ 1044. Brno: 10.1. 2004

[5] Tůma, J. Zpracování signálů získaných z mechanických systémů užitím FFT „1.vydání“, Sdělovací technika, Praha 1997, ISBN 80-901936-1-7

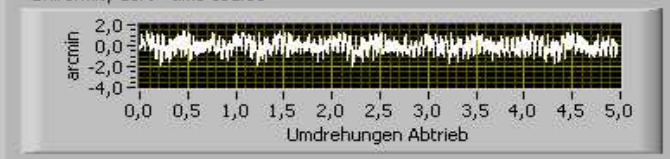
[6] http://www.harmonicdrive.net/reference/operatingprinciples/

[7] National Instruments. Database Connectivity Toolset User Manual. May 2001 - zdroj http://www.ni.com/pdf/manuals/321525c.pdf

SEZNAM PŘÍLOH

Uniformity left | Uniformity right | Lost Motion | Uniformity left - Probability | Uniformity right - Probability | Lost Motion - Probability

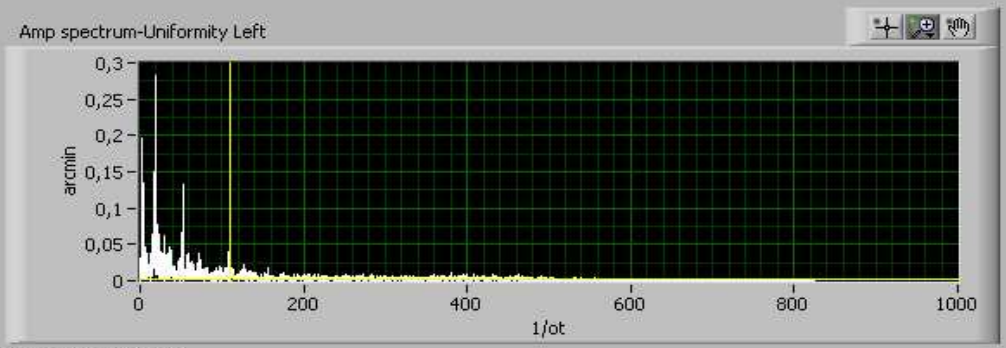
Uniformity Left - time course



Wanted frequencies

	1.stupeň	2.stupeň	3.stupeň
f0	20,00	0,00	4,00
f1	4,00	0,00	1,00
f2	12,00	0,00	3,00
f3	48,00	0,00	9,00
f4	16,00	0,00	3,00
f5	432,00	0,00	108,00
f6	355,38	0,00	324,00
f7	432,00	0,00	108,00

Amp spectrum-Uniformity Left



V tabulce nad tímto textem jsou vypočítané hledané frekvence. Pokud jsou ve sloupci samé nuly, znamená to, že převodovka daný stupeň neobsahuje.
V tabulkách uniformity left a uniformity righht jsou ve sloupci notes naznačeny frekvence, které se shodují s vyhodnocenými.
Formát těchto poznámek je ve tvaru - fxy
- x značí stupeň (1 znamená 1.stupeň)
- y značí danou frekvenci (fx2 znamená f2 pro příslušný stupeň

Vysvětlení hledaných frekvencí
f0 - otáčení centrálního kola, tj.vstupního hřídele daného stupně
f1 - otáčení unášeče, tj.výstupního hřídele daného stupně
f2 - dotyk zvoleného zubu korunového kola s planetovými koly daného stupně
f3 - dotyk zvoleného zubu centrálního kola s planetovými koly daného stupně
f4 - dotyk zvoleného zubu planetového kola buď s korunovým kolem nebo s centrálním kolem daného stupně
f5 - frekvence dotyku zubů korunového kola se zuby jednoho planetového kola daného stupně
f6 - frekvence dotyku zubů korunového kola současně se všemi planetovými koly daného stupně
f7 - frekvence dotyku zubů korunového kola postupně se všemi planetovými koly daného stupně

Table - uniformity left

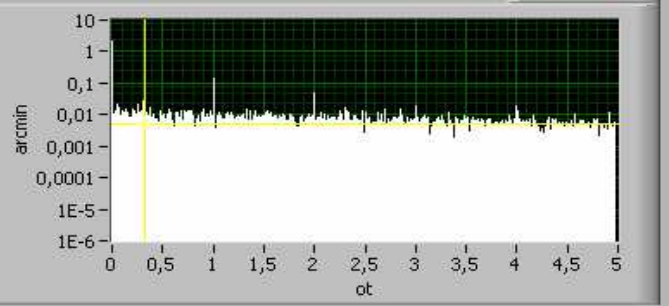
number	1/ot	amplitude	phase[rad]	notes
1	18,9634	0,2828	22,2479	19.f31
2	2,0138	0,1960	11,9792	2.f31
3	52,0104	0,1315	-11,1040	13.f11 13.f30 52.f31
4	15,9710	0,1268	25,1848	4.f11 1.f14 4.f30 16.f31
5	3,0400	0,1228	10,5678	3.f31 1.f32 1.f34
6	21,9788	0,0763	30,4176	22.f31
7	17,9166	0,0645	22,9933	2.f33
8	16,9438	0,0621	22,0203	17.f31
9	29,0258	0,0610	17,9061	29.f31
10	22,9804	0,0549	21,5633	23.f31
11	20,9845	0,0514	27,7773	21.f31 7.f32 7.f34
12	6,0113	0,0472	13,1623	6.f31 2.f32 2.f34
13	35,0469	0,0466	1,9179	35.f31
14	38,9512	0,0434	2,9501	39.f31 13.f32 13.f34
15	108,0490	0,0400	-46,3392	27.f11 9.f12 27.f30 108.f31 36.f32 12.f33 36.f34 1.f35 1.f37
16	60,0720	0,0364	-24,7430	3.f10 5.f12

Uniformity left | Uniformity right | Lost Motion | Uniformity left - Probability | Uniformity right - Probability | Lost Motion - Probability

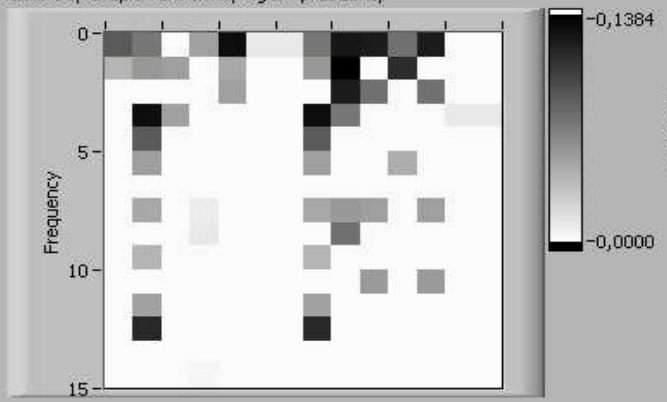
Probability - uniformity right

	f10	f11	f12	f13	f14	f15	f17	f30	f31	f32	f33	f34	f35
1.	0,0886	0,0729	0	0,0497	0,1310	0,0100	0,0100	0,0729	0,1263	0,1234	0,0765	0,1234	0
2.	0,0394	0,0543	0,0507	0	0,0461	0	0	0,0543	0,1384	0	0,1147	0	0
3.	0	0	0	0	0,0497	0	0	0	0,1234	0,0765	0	0,0765	0
4.	0	0,1310	0,0497	0	0	0	0	0,1310	0,0729	0	0	0	0,0100
5.	0	0,0886	0	0	0	0	0	0,0886	0	0	0	0	0
6.	0	0,0507	0	0	0	0	0	0,0507	0	0	0,0425	0	0
7.	0	0	0	0	0	0	0	0	0	0	0	0	0
8.	0	0,0461	0	0,0084	0	0	0	0,0461	0,0543	0,0507	0	0,0507	0
9.	0	0	0	0,0100	0	0	0	0	0,0765	0	0	0	0
10.	0	0,0394	0	0	0	0	0	0,0394	0	0	0	0	0
11.	0	0	0	0	0	0	0	0	0	0,0534	0	0,0534	0
12.	0	0,0497	0	0	0	0	0	0,0497	0	0	0	0	0
13.	0	0,1168	0	0	0	0	0	0,1168	0	0	0	0	0
14.	0	0	0	0	0	0	0	0	0	0	0	0	0
15.	0	0	0	0,0028	0	0	0	0	0	0	0	0	0

Real cepstrum-Uniformity Right



Intensity Graph - uniformity right - probability



Tato záložka slouží k přibližnému určení pravděpodobnosti přiřazení jednotlivých harmonických složek k jednotlivým frekvencím, které jsou popsány na předchozích záložkách.

Jednotlivé složky frekvenčního spektra mohou být harmonickou nejenom jedné frekvence. K určení, ke které frekvenci se více hodí (jsou tedy harmonickou určité frekvence s vyšší pravděpodobností než jiné frekvence), má sloužit tabulka a také graf pod ní. V tabulce jsou vypsány hodnoty amplitud jednotlivých složek (od 1. do 15. harmonické) od dané frekvence, ke které se našla alespoň jedna složka ve frekvenčním spektru. Potom je více pravděpodobné, že nalezená složka patří k první frekvenci než ke druhé, pokud zároveň s touto složkou byly nalezeny další složky k první frekvenci, zatímco ke druhé frekvenci byla nalezena pouze tato jedna složka.

Graf zobrazený pod tabulkou má napomoci vizuální představě. Hodnoty amplitud jsou zde zobrazeny v černobílém spektru s tím, že nula je zobrazena jako bílá a maximální amplituda jako černá.

Graf napravo zobrazuje reálné kepstrum naměřeného průběhu. Byla snaha z tohoto grafu určovat pravděpodnost, ale nakonec to nevedlo k žádanému výsledku.